# ΒΕΛΤΙΣΤΕΣ ΤΡΙΓΩΝΟΠΟΙΗΣΕΙΣ ΚΥΡΤΩΝ ΠΟΛΥΓΩΝΩΝ

Η
ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ

Υποβάλλεται στην

ορισθείσα από την Γενική Συνέλευση Ειδικής Σύνθεσης
του Τμήματος Πληροφορικής
Εξεταστική Επιτροπή

από την

Λαμπρινή Καμωνά

ως μέρος των Υποχρεώσεων

για τη λήψη

του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

ΜΕ ΕΞΕΙΔΙΚΕΥΣΗ ΣΤΗΝ ΘΕΩΡΙΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Ιούλιος 2008

# ΠΕΡΙΕΧΟΜΕΝΑ

# ΕΥΡΕΤΗΡΙΟ ΠΙΝΑΚΩΝ

# ΕΥΡΕΤΗΡΙΟ ΣΧΗΜΑΤΩΝ

# ΠΕΡΙΛΗΨΗ

Λαμπρινή Καμωνά του Κωνσταντίνου και της Ελένης. MSc, Τμήμα Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, Ιούλιος, 2008. Βέλτιστες Τριγωνοποιήσεις Κυρτών Πολυγώνων. Επιβλέποντας: Λεωνίδας Παληός.

Σε αυτή την εργασία μελετώνται τριγωνοποιήσεις κυρτών πολυγώνων, δηλαδή, διαμερίσεις πολυγώνων με χρήση διαγωνίων, οι οποίες είναι βέλτιστες ως προς κάποια κριτήρια ποιότητας. Ο γενικός αλγόριθμος για τον υπολογισμό των τριγωνοποιήσεων βασίζεται στην τεχνική του δυναμικού προγραμματισμού ενώ ως κριτήρια έχουν χρησιμοποιηθεί:

- το εμβαδόν των τριγώνων
- το μήκος της ακτίνας του εγγεγραμμένου κύκλου του τριγώνου
- το μήκος της ακτίνας του περιγεγραμμένου κύκλου του τριγώνου
- ο λόγος του μήκους της ακτίνας του περιγεγραμμένου κύκλου του τριγώνου προς το μήκος της ακτίνας του εγγεγραμμένου κύκλου του τριγώνου
- η γωνία του τριγώνου.

Η υλοποίηση υπολογίζει τριγωνοποιήσεις που μεγιστοποιούν την ελάχιστη τιμή (*Max_Min*) ή ελαχιστοποιούν τη μέγιστη τιμή (*Min_Max*) καθενός από τα παραπάνω κριτήρια.

Επίσης, αποδόθηκαν γραφικά οι τιμές επιμέρους λύσεων για τα παραπάνω κριτήρια ώστε να παρατηρηθούν ιδιότητες που θα βοηθούσαν στην περιγραφή ταχύτερων αλγορίθμων.

# EXTENDED ABSTRACT IN ENGLISH

Kamona, Lamprini, KL. MSc, Computer Science Department, University of Ioannina, Greece. July, 2008. Optimal Triangulations of Convex Polygons. Thesis Supervisor: Leonidas Palios.

In this project it is studied Triangulations of Convex Polygons that are Optimal in some quality criteria. Saying Triangulations we mean the partitioning of the polygon into triangles, using diagonals among pairs of vertices.

At first there are mentioned some of the fundamental principles of geometry as far as our work is concerned, including geometric concepts, regulations and equations. Therefore there are mentioned the concepts of Polygons, Convexity of Polygons, the Circumscribed and Inscribed Circle of a Triangle and their relationship with the triangle's vertices. There is a useful equation that is used to calculate a triangle's area, which uses only the x-, y- coordinates of a triangle and is given by an determinant of its vertices coordinates.

More, we were motivated in analyzing some of the most common quality measures of a triangle. That is:

- The ratio, of the radius of the circumscribed to that of the inscribed circle, known as radii-ratio ($\rho = R / r$)
- The ratio, of the largest length to that of the smallest length of an edge of a triangle, known as the edge-ratio, $\tau$.
- The ratio of the radius of the circumscribed circle to that of the edge of the minimum length or the maximum length of a triangle.
- The ratio of the radius of the circumscribed circle to the half-perimeter of a triangle ($\nu = R / p$)
- The ratio of the radius of the inscribed circle to the half-perimeter of a triangle ($r / p$).

It is said the quality measures of $\rho$, $\nu$ and $r/p$ have equivalent results and are those measures that can distinguish good quality triangles [12].

Furthermore, we mention general triangulations algorithms with their corresponding time complexities. Moreover, we refer to some of the most well known Optimal Triangulations of Convex Polygons, such as Delaunay Triangulation, Minimum Weight Triangulation, Edge Insertion Technic Triangulations and a recent triangulation algorithm, which is optimal for the criterion of the Area of a Triangle referring to it, with the name of its publishers authors, which are J.M. Keil and T. S. Vassilev.

Keil's and Vassilev's optimal triangulation algorithm was one of the motivating factors to analyze their work and try to extend their results to triangulations including other criteria such as angle, radii-ratio and other. Thus, we analyzed their work in calculating the Max_Min and Min_Max Area Triangulation Algorithms taking in notice the proofs of the lemmas and definitions of new concepts, regarding convex polygons, such as zonality of a polygon, unimodality of distance functions and area of triangles configured by an base edge and a third vertex lying at the boundary of P. These observations and other are of great importance in demonstrating that their time complexity is of $O(n^2 \log n)$ and space complexity of $O(n^2)$, together with the use of appropriate structures, such as balanced search binary trees and staircase structures.

Moreover, our concern was in implementing a general algorithm, which uses the primitives of the dynamic programming in order to find optimal triangulations in the criteria of:

- Area
- Inradius
- Circumradius
- Radii-ratio and
- Angle.

Our Optimization algorithm runs in both the Max_Min and the Min_Max form in order to minimize the maximum value or to maximize the minimum value among all possible triangulations of the polygon P. Our algorithm has $O(n^3)$ time complexity and $O(n^2)$ space complexity.

We tested the results of our algorithm in various polygons with varying criteria and optimization type in order to conclude in some conclusions. The one thing that we concluded is that our results agree with that of the Keil's and Vassilev's and some future work should concern in answering further issues.
Some of our future work should relate with the expansion of the algorithm regarding simple polygons and larger randomized polygons.

# ΚΕΦΑΛΑΙΟ 1. ΕΙΣΑΓΩΓΗ



## 1.1. Στόχοι

Ο συνολικός στόχος της διατριβής είναι να μελετήσει βέλτιστες τριγωνοποιήσεις κυρτών πολυγώνων ως προς διάφορα κριτήρια ποιότητας ενός τριγώνου και να εξάγει κάποια συμπεράσματα ως προς τις ιδιότητες των λύσεων έτσι ώστε σε μελλοντικές επεκτάσεις να εξαχθούν αποτελεσματικότεροι, ταχύτεροι αλγόριθμοι, ως προς τα ίδια κριτήρια.

## 1.2. Δομή της Διατριβής

Η διατριβή περιέχει 4 κεφάλαια: Το Κεφάλαιο 1 περιλαμβάνει την Εισαγωγή, όπου αναφέρονται οι στόχοι της εργασίας και παρουσιάζεται η δομή της διατριβής. Το Κεφάλαιο 2 παρουσιάζει κάποιες βασικές γεωμετρικές έννοιες και κάποια βασικά στοιχεία της Τριγωνοποίησης, όπως τα μέτρα ποιότητας τριγώνων και υπολογισμούς τριγωνοποιήσεων. Το Κεφάλαιο 3 παρουσιάζει τις βασικές αρχές του δυναμικού προγραμματισμού, τον γενικό αλγόριθμο της λύσης και κάποιους ενδεικτικά βέλτιστους αλγορίθμους τριγωνοποίησης, όπως ο αλγόριθμος των Keil & Vassilev, η Delaunay διαίρεση σε τρίγωνα και η Minimum Weight Τριγωνοποίηση. Το Κεφάλαιο 4 παρουσιάζει την Υλοποίηση και κάποια Πειραματικά Αποτελέσματα, όπου παρουσιάζονται η είσοδος και η έξοδος του αλγορίθμου, κάποιες δομές, επιλεγμένες συναρτήσεις και ενδεικτικές εκτελέσεις. Τέλος, το Κεφάλαιο 5 παρουσιάζει τα συμπεράσματα και τις πιθανές επεκτάσεις της εργασίας.

# ΚΕΦΑΛΑΙΟ 2. ΕΙΣΑΓΩΓΙΚΕΣ ΕΝΝΟΙΕΣ



Στο κεφάλαιο αυτό θα παρουσιασθούν βασικές γεωμετρικές έννοιες της υπολογιστικής γεωμετρίας και κάποια στοιχεία για τη διαδικασία χωρισμού ενός πολυγώνου σε τρίγωνα, γνωστή με τον όρο τριγωνοποίηση. Θα μιλήσουμε επίσης για μέτρα ποιότητας ενός τριγώνου. Οι ενότητες που θα ακολουθήσουν είναι οι Γεωμετρικές Έννοιες, που αναφέρεται σε βασική θεωρία πολυγώνων, στον υπολογισμό εμβαδού ενός τριγώνου, στον περιγεγραμμένο και εγγεγραμμένο κύκλο τριγώνου

## 2.1. Γεωμετρικές Έννοιες

Οι γεωμετρικές έννοιες που θα αναλυθούν είναι η έννοια του πολυγώνου, της κυρτότητας ενός συνόλου σημείων, η έννοια του εμβαδού ενός τριγώνου, όπως επίσης και η έννοια του εγγεγραμμένου και του περιγεγραμμένου κύκλου ενός τριγώνου και η σχέση τους με τα στοιχεία του τριγώνου.

### *2.1.1. Πολύγωνα*

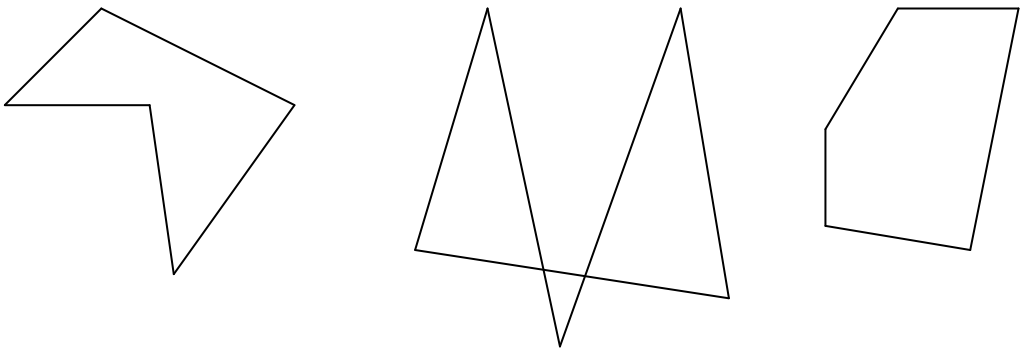
Τα πολύγωνα χρησιμοποιούνται για να προσεγγίσουν την αναπαράσταση διάφορων αντικειμένων του πραγματικού κόσμου καθώς μπορούν να περιγράψουν αρκετά αντικείμενα με ακρίβεια και ο χειρισμός τους στον υπολογιστή είναι εύκολος. Έτσι, μπορούν να χρησιμοποιηθούν για να περιγράψουν επιφάνειες ή τμήματα επιφανειών αντικειμένων, προκειμένου να αναπαρασταθούν γραφικά στην οθόνη ενός υπολογιστή, να περιγράψουν σχηματικά την περιοχή των εμποδίων ενός ρομπότ, προκειμένου να κινηθεί ανάλογα ή ακόμα και σχήματα γραμμάτων για τη χρήση τους σε συστήματα αυτόματης αναγνώρισης γραφής από τον υπολογιστή.

Παρακάτω θα αναλυθεί πώς ορίζεται μαθηματικά ένα πολύγωνο και ποιοι είναι οι διάφοροι διαχωρισμοί του.

Αρχικά θα οριστεί η έννοια του απλού πολυγώνου. Ένα *απλό πολύγωνο* (s*imple polygon*) είναι η περιοχή του επιπέδου που περικλείεται από ένα *πεπερασμένο* σύνολο ευθυγράμμων τμημάτων, τα οποία σχηματίζουν μια *απλή κλειστή πολυγωνική* γραμμή. *Πολυγωνική* γραμμή χαρακτηρίζεται γιατί τα ευθύγραμμα τμήματα είναι συνδεδεμένα στη σειρά, το ένα μετά το άλλο. *Κλειστή*, γιατί τα τμήματα σχηματίζουν ένα κύκλο και *απλή*, γιατί μή διαδοχικά τμήματα δεν τέμνονται.

Μια άλλη κατηγορία πολυγώνων, που ανήκει στα απλά πολύγωνα είναι τα *κυρτά* πολύγωνα. Πρόκειται για πολύγωνα, όπου για κάθε ζεύγος σημείων *p, q* του πολυγώνου το ευθύγραμμο τμήμα *pq* που τα συνδέει ανήκει πλήρως στο πολύγωνο.

Τα σημεία που τέμνονται τα ευθύγραμμα τμήματα καλούνται *κορυφές* (*vertices*) και τα ευθύγραμμα τμήματα, που συνδέουν δύο διαδοχικές κορυφές, ονομάζονται ακμές (*edges)* του πολυγώνου.



(α) Απλό πολύγωνο (β) Όχι Απλό Πολύγωνο (γ) Κυρτό Πολύγωνο

Σχήμα 2-1 Τρία Παραδείγματα Πολυγώνων

*2.1.2. Εμβαδόν τριγώνου*

Ας θεωρήσουμε ένα τρίγωνο ΑΒΓ με κορυφές τα σημεία *Α* με συντεταγμένες ($\chi_{A,}$ $y_A$), *Β* με συντεταγμένες ($\chi_{B,}$ $y_B$) και *Γ* με συντεταγμένες ($X_{\Gamma,}$ $Y_\Gamma$). Τα μήκη των πλευρών δηλώνονται αντίστοιχα ως *a* το μήκος της πλευράς ΒΓ, *b* το μήκος της πλευράς ΑΓ και *c* το μήκος της πλευράς ΑΒ και αντίστοιχα και οι γωνίες ως *α, β* και *γ*.

Τα μήκη των πλευρών *a, b* και *c* του τριγώνου δίνονται από τους παρακάτω τύπους:

| | |
|---|---|
| $a = \sqrt{(y_\Gamma - y_B)^2 + (x_\Gamma - x_B)^2}$ | Εξ. 2.1 |
| $b = \sqrt{(y_\Gamma - y_A)^2 + (x_\Gamma - x_A)^2}$ | |
| $c = \sqrt{(y_B - y_A)^2 + (x_B - x_A)^2}$ | |

Είναι χρήσιμο να επισημανθεί σε αυτό το σημείο ότι η διάταξη των γωνιών του τριγώνου ακολουθεί τη διάταξη των απέναντι τους ακμών στο τρίγωνο. Ισχύει δηλαδή, ότι, εάν

$a < b < c$,

τότε ισχύει και ότι

$\alpha < \beta < \gamma$ .

Το εμβαδόν του τριγώνου ΑΒΓ ορίζεται να είναι ίσο με:

$$E_{ABΓ} = \frac{1}{2}\ AΓ\ BΔ = \frac{1}{2}\ BΓ\ AZ = \frac{1}{2}\ AB\ ΓH\ ,$$

όπου Z, Η και Δ είναι τα ίχνη από τα ύψη των κορυφών Α, Β και Γ προς τις ακμές ΒΓ, ΒΑ και ΑΓ.

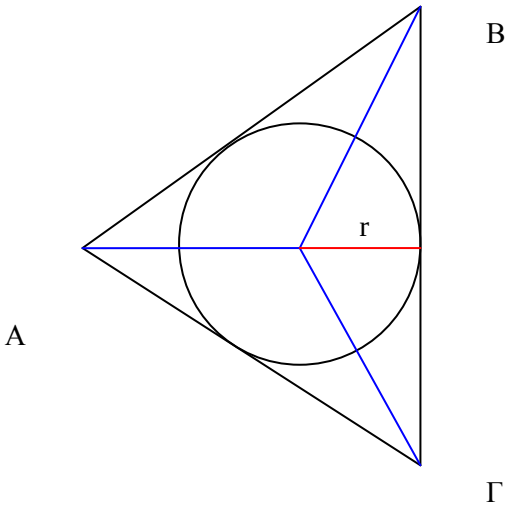
Ένας άλλος τύπος υπολογισμού, ιδιαίτερα χρήσιμος σε αλγορίθμους υπολογιστικής γεωμετρίας για τον υπολογισμό του εμβαδού ενός τριγώνου είναι ο παρακάτω:

$$E_{ABC} = \frac{1}{2}\begin{vmatrix} \chi_A & y_A & 1 \\ \chi_B & y_B & 1 \\ \chi_C & y_C & 1 \end{vmatrix} = \frac{1}{2}\ (\chi_A\ y_B - y_A\ \chi_B + y_A\ \chi_C - \chi_A\ y_C + \chi_B\ y_C - y_B\ \chi_C). \qquad \text{Εξ. 2.2}$$

Ο τρόπος αυτός είναι πιο εύχρηστος στην υλοποίηση γιατί στηρίζεται μόνο στις συντεταγμένες των κορυφών του τριγώνου.

*2.1.3. Εγγεγραμμένος Κύκλος τριγώνου*

Ο εγγεγραμμένος κύκλος ενός τριγώνου, όπου απεικονίζεται σχηματικά παρακάτω, έχει το χαρακτηριστικό ότι η ακτίνα του, *r*, εκφράζει ένα μέτρο της ποιότητας του τριγώνου, όπως θα αναλυθεί παρακάτω.




Σχήμα 2-2 Εγγεγραμμένος Κύκλος Τριγώνου

Όπως προκύπτει και από το σχήμα, το κέντρο του εγγεγραμμένου στο τρίγωνο κύκλου είναι το σημείο τομής των διχοτόμων των γωνιών του. Και υπολογίζεται μέσα από τον παρακάτω τύπο της βασικής γεωμετρίας, που είναι :

$$E = r * p, \qquad \text{Εξ. 2.3}$$

όπου $p$ η ημιπερίμετρος του τριγώνου, δηλαδή:

$$p = \frac{a+b+c}{2}. \qquad \text{Εξ. 2.4}$$

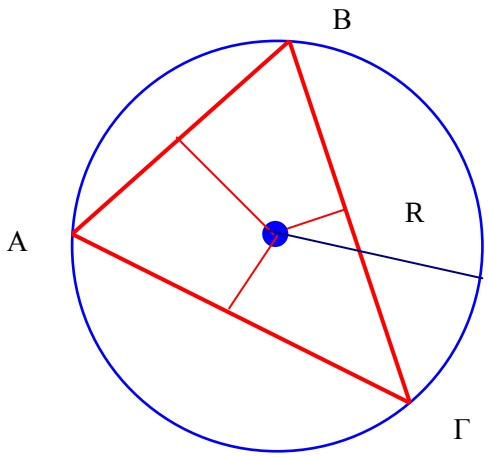
Συνεπώς,

$$r = \frac{2E}{a+b+c}. \qquad \text{Εξ. 2.5}$$

Καθώς τα μήκη των πλευρών $a$, $b$ και $c$ μπορούν να εκφραστούν συναρτήσει των συντεταγμένων των τριών κορυφών Α, Β και Γ (Εξ. 2.1) και όπως ειπώθηκε και ο υπολογισμός του εμβαδού $E$ (Εξ. 2.2) εκφράζεται συναρτήσει των συντεταγμένων των τριών κορυφών του, άρα και το $r$ μπορεί να υπολογιστεί γνωρίζοντας μόνο τις συντεταγμένες των κορυφών του τριγώνου σε σταθερό χρόνο.

### *2.1.4. Περιγεγραμμένος Κύκλος τριγώνου*

Όσον αφορά τον περιγεγραμμένο κύκλο του τριγώνου (Σχήμα 1.4) το κέντρο του κύκλου βρίσκεται στο σημείο τομής των μεσοκαθέτων των τριών πλευρών του τριγώνου και συμπίπτει με τη μοναδική κορυφή του διαγράμματος Voronoi των τριών κορυφών του τριγώνου.




Σχήμα 2-3 Περιγεγραμμένος Κύκλος Τριγώνου

Για τον υπολογισμό του μήκους της ακτίνας, R, του περιγεγραμμένου κύκλου του τριγώνου, θεωρούμε τον τύπο από τη βασική γεωμετρία, που είναι ο ακόλουθος:

$$2\,R = \frac{abc}{2E}$$ Εξ. 2.4

από όπου προκύπτει για το R, ότι

$$R = \frac{abc}{4E}$$ Εξ. 2.5

Με αυτό τον τρόπο μπορούμε να εκφράσουμε και την ακτίνα του περιγεγραμμένου κύκλου συναρτήσει των συντεταγμένων των τριών κορυφών του.

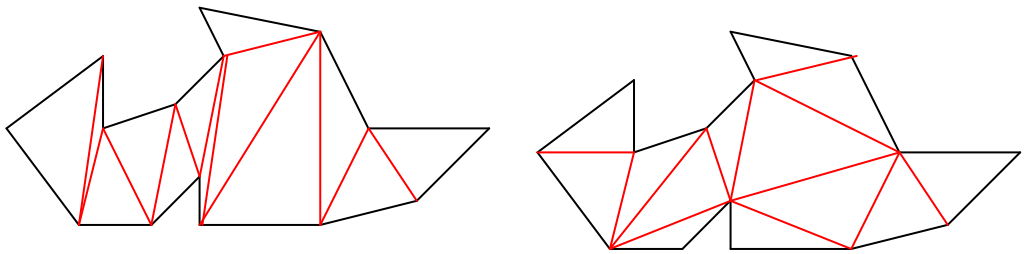
### 2.2. Τριγωνοποίηση

Επειδή το πολύγωνο μπορεί να αποτελέσει μια αρκετά πολύπλοκη δομή, κρίνεται χρήσιμος ο διαχωρισμός του σε απλούστερα κομμάτια. Ιδιαιτέρα χρήσιμη είναι η διαίρεση ενός πολυγώνου σε τρίγωνα.

Η διαίρεση ενός πολυγώνου σε τρίγωνα προκύπτει με την εισαγωγή διαγωνίων (diagonals) ανάμεσα σε ζευγάρια κορυφών, έτσι ώστε κανένα ζευγάρι διαγωνίων να μην τέμνεται μεταξύ του. Έτσι κάθε τρίγωνο, που προκύπτει έχει σαν πλευρές του ακμές ή διαγώνιους του πολυγώνου. Τρίγωνα, που

έχουν κορυφές στο εσωτερικό τους δεν είναι αποδεκτά και μπορεί να προκύψουν, όταν το πολύγωνο έχει τρεις συνευθειακές κορυφές.

Παρακάτω παρουσιάζονται σχηματικά δύο διαφορετικοί χωρισμοί σε τρίγωνα του ίδιου πολυγώνου.



Σχήμα 2-4 Δύο Διαφορετικές Τριγωνοποιήσεις του Ίδιου Πολυγώνου

Ο αριθμός των τριγώνων στα οποία διαιρείται ένα σύνολο σημείων, *S*, είναι ίσος με *2n-h-2* τρίγωνα, όπου *n* είναι το πλήθος των σημείων και *h* τα σημεία στο εσωτερικό του κυρτού περιβλήματος των σημείων και οι ακμές είναι 3*n-h*-3.

Παρακάτω θα παρουσιασθεί το πλήθος των διαφορετικών χωρισμών σε τρίγωνα ενός πολυγώνου. Προκύπτει ότι η διαίρεση ενός πολυγώνου σε τρίγωνα δεν είναι μοναδική και ότι ο αριθμός των διαφορετικών διαιρέσεων σε τρίγωνα ενός πολυγώνου είναι εκθετικός στο πλήθος των κορυφών του πολυγώνου. Μάλιστα ισχύει ότι το πλήθος των διαφορετικών χωρισμών σε τρίγωνα ενός απλού πολυγώνου *n+2* πλευρών είναι ίσος με τον *Catalan* αριθμό, που υπολογίζεται από τη σχέση

$$C_n = \binom{2n}{n} / (n+1)$$

και είναι της τάξης $\Omega ( 4^n / n^{3/2})$.

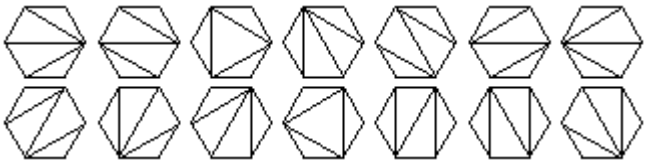
Για τιμές του *n*, $0 \le n \le 10$ οι αριθμοί *Catalan* δίνονται από τον παρακάτω πίνακα:

Πίνακας 2.1 Οι 10 πρώτοι αριθμοί Catalan

| n | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $C_n$ | 1 | 1 | 2 | 5 | 14 | 42 | 132 | 429 | 1430 | 4862 | 16796 |

Οι ίδιοι αριθμοί Catalan υπολογίζουν επίσης το πλήθος των δυαδικών δένδρων με *n*+1 φύλλα.

Για ένα παράδειγμα κυρτού πολυγώνου 6 πλευρών οι διαφορετικοί χωρισμοί που προκύπτουν είναι 14 (=$C_4$) στο πλήθος και παρουσιάζονται σχηματικά στο Σχήμα 2.5.



Σχήμα 2-5 6 πλευρές, 14 διαφορετικοί χωρισμοί

*2.2.1. Αλγόριθμοι για τριγωνοποίηση πολυγώνου*

Στην παράγραφο αυτή αναλύονται αλγόριθμοι χωρισμού απλού πολυγώνου σε τρίγωνα που δεν θεωρούν κριτήρια βελτιστοποίησης. Οι πρώτοι αλγόριθμοι χωρισμού σε τρίγωνα απλού πολυγώνου ήταν τετραγωνικοί. Ο πρώτος αλγόριθμος με χρονική πολυπλοκότητα O(*n* log*n*) δημοσιεύτηκε το 1978. Ακολούθησαν αλγόριθμοι μικρότερης πολυπλοκότητας για ειδικές όμως περιπτώσεις. Ο αλγόριθμος των *Hertel* και *Mehlhorn*, πολυπλοκότητας O(*n* + *r* log *r*), όπου *r* το πλήθος των μη κυρτών κορυφών. Ο αλγόριθμος των *Chazelle* και *Incerpi*, πολυπλοκότητας O(*n* log *s*), με *s* τη *sinuosity* - ένα μέτρο μη κυρτότητας του πολυγώνου.

Στη χειρότερη περίπτωση, όλοι αυτοί οι αλγόριθμοι απαιτούν O( *n* log *n*) χρόνο.

Το 1988, οι *Tarzan* και *VanWyk* περιέγραψαν έναν O(*n*log( log *n*)) αλγόριθμο, ενώ ακολούθησαν δύο O( *n*log**n*) αλγόριθμοι. Τέλος, το 1991 προέκυψε από τον *Chazelle* ένας γραμμικός αλγόριθμος.

**2.3. Μέτρα ποιότητας**

Για την αποτίμηση της ποιότητας ενός τριγώνου, το κατά πόσο «καλό» ή όχι είναι ένα τρίγωνο έχουν καθοριστεί κάποια μετρήσιμα στοιχεία. Συγκεκριμένα, ως *μέτρο ποιότητας* (*quality measure*) είναι μια συνάρτηση ή ένας συνδυασμός συναρτήσεων, που ορίζει το πόσο καλή είναι μια τριγωνοποίηση, προκειμένου να συγκριθούν διαφορετικές τριγωνοποιήσεις.

Τα αναμενόμενα κριτήρια που αντανακλούν την «ποιότητα» ενός τριγώνου είναι συνήθως οι γωνίες του, τα μήκη των πλευρών του ή το εμβαδόν του.

Για τα επόμενα, που θα ακολουθήσουν, συμβολίζουμε ως $\theta_0$ τη μικρότερη γωνία και ως $\theta_\infty$ τη μεγαλύτερη γωνία ενός τριγώνου και για τις οποίες ισχύει ότι:

$$0 < \theta_0 \leq \frac{\pi}{3} \leq \theta_\infty$$

και το άθροισμα των γωνιών του τριγώνου είναι ίσο με π, δηλαδή $\alpha + \beta + \gamma = \pi$.

Τα πιο γνωστά και χρησιμοποιήσιμα μέτρα ποιότητας είναι τα εξής:

**1.** Ο λόγος της ακτίνας του περιγεγραμμένου κύκλου του τριγώνου, R, προς την ακτίνα, r, του εγγεγραμμένου κύκλου του τριγώνου, γνωστό ως *radii ratio* ( ρ).

$$\rho = \frac{R}{r} \qquad \text{Εξ. 2.6}$$

Συνδυάζοντας τις σχέσεις (2.3), (2.5) και (2.6) η παραπάνω ισότητα μετατρέπεται στην:

$$\rho = \frac{abc(a+b+c)}{8E^2} \qquad \text{Εξ. 2.7}$$

όπου το εμβαδόν του τριγώνου μπορεί να προκύψει από τον υπολογισμό της παρακάτω ορίζουσας:

$$E_{ABC} = \frac{1}{2}\begin{vmatrix} X_A & Y_A & 1 \\ X_B & Y_B & 1 \\ X_C & Y_C & 1 \end{vmatrix} \frac{1}{2}\ (X_A\ Y_B - Y_A\ X_B + Y_A\ X_C - X_A\ Y_C + X_B\ Y_C - Y_B\ X_C).$$

Άρα, ο λόγος ρ εκφράζεται συναρτήσει των συντεταγμένων των κορυφών του.

Παρακάτω θα αναλυθεί το πώς μεταβάλλεται η ποιότητα ενός τριγώνου συγκριτικά με τη μεταβολή της τιμής του λόγου ρ.

Για την εξαγωγή αυτού του συμπεράσματος εκφράζεται ο λόγος ρ ως

$$\rho = \frac{\sin a + \sin b + \sin(a+b)}{2\sin a \sin b \sin(a+b)} \qquad \text{Εξ. 2.8}$$

,όπου αν θέσουμε $p = \tan\frac{a}{2}$ και $q = \tan\frac{b}{2}$, τότε για το λόγο ρ προκύπτει η ισότητα:

$$\rho = \frac{(1+p^2)(1+q^2)}{4p*q*(1-p*q)} \qquad \text{Εξ. 2.9}$$

Ο λόγος αυτός είναι πάντα θετικός, όσο το γινόμενο p*q είναι μικρότερο της μονάδας, δηλαδή p* q < 1. Κάτι τέτοιο ισχύει εφόσον για τις γωνίες του τριγώνου ισχύει ότι: $\alpha + \beta + \gamma \leq \pi$.

Ο μέγιστος λόγος σύμφωνα με τον τύπο (2.9) επιτυγχάνεται στο σημείο, όπου η πρώτη παράγωγος είναι ίση με μηδέν και προκύπτει όταν το τρίγωνο είναι ισόπλευρο. Τότε ισχύει για την τιμή του ρ, ότι: $\rho = 2$ ,
ή ισοδύναμα ότι
$R = 2\,r$ Εξ. 2.10

Δηλαδή ο μέγιστος λόγος επιτυγχάνεται, όταν το μήκος της ακτίνας του περιγεγραμμένου κύκλου είναι διπλάσιο εκείνου της ακτίνας του εγγεγραμμένου κύκλου του τριγώνου.

**2.** Ο λόγος της μεγίστου προς εκείνης του ελαχίστου μήκους ακμής ενός τριγώνου (τ, *edge ratio*).

$\tau = |t_\infty| / |t_0|$ , όπου $|t_0|$, $|t_\infty|$ τα μήκη της μεγαλύτερης και της μικρότερης ακτίνας.

Επομένως ο λόγος αυτός είναι πάντα μεγαλύτερος ή ίσος της μονάδας. Η ισότητα ισχύει στη περίπτωση που $|t_0| = |t_\infty|$, δηλαδή τα μήκη και των τριών πλευρών είναι ίσα, οπότε το τρίγωνο είναι ισόπλευρο (equilateral).

Όπως και στη περίπτωση του ρ, η ελάχιστη τιμή του εντοπίζεται στη περίπτωση που το τρίγωνο είναι ισόπλευρο. Η συνάρτηση για τις δεδομένες τιμές των γωνιών του τριγώνου δεν έχει στάσιμο σημείο. Σαν συνάρτηση των γωνιών μπορεί να εκφραστεί με το λόγο

$$\tau = \frac{\sin\theta_\infty}{\sin\theta_0} \qquad \text{Εξ. 2.11}$$

Αποδεικνύεται ότι η συνάρτηση $\theta_0 \rightarrow \tau(\theta_0, \theta_\infty)$ είναι συνάρτηση φθίνουσα για οποιαδήποτε τιμή της $\theta_\infty$. Όπου $\theta_0$, $\theta_\infty$ είναι η μικρότερη και η μέγιστη γωνία του τριγώνου και σύμφωνα με μια ανακατανομή των δεικτών των κορυφών μπορεί να είναι οποιεσδήποτε από τις γωνίες του τριγώνου.

**3.** Ο λόγος της ακτίνας του περιγεγραμμένου κύκλου προς το μέγιστο μήκος της ακμής ενός τριγώνου (*edge to circumradius*).

Από τη σχέση

$$2R = \frac{a}{\sin\alpha} = \frac{b}{\sin\beta} = \frac{c}{\sin\gamma} \qquad \text{Εξ. 2.12}$$

και του γεγονότος ότι η διάταξη των γωνιών ακολουθεί τη διάταξη των αντιστοίχων τους απέναντι πλευρών προκύπτουν οι παρακάτω ισότητες:

$$\frac{R}{|t|_\infty} = \frac{1}{2\sin\theta_\infty} \text{ και } \frac{R}{|t|_0} = \frac{1}{2\sin\theta_0}.$$

Και για το μέτρο αυτό προκύπτει ότι συναντάει την ελάχιστη τιμή του στη περίπτωση του ισόπλευρου τριγώνου. Πρόκειται για μια φθίνουσα συνάρτηση.

**4.** Ο λόγος της ακτίνας του εγγεγραμμένου κύκλου προς το ελάχιστο μήκος της ακμής ενός τριγώνου (*edge to inradius*).

Ισχύει ότι

$$\nu = \frac{R}{p} = \frac{2R}{a+b+c} = \frac{1}{\sin a + \sin b + \sin\gamma} \qquad \text{Εξ. 2.13}$$

Η (2.13) σε συνδυασμό με την (2.10) δίνει:

$$\frac{p}{r} = \frac{\rho}{\nu} = \frac{(\sin a + \sin\beta + \sin\gamma)^2}{2\sin a \sin\beta \sin\gamma} = \frac{(\sin\theta_0 + \sin\theta_\infty + \sin(\theta_0+\theta_\infty))^2}{2\sin\theta_0 \sin\theta_\infty \sin(\theta_0+\theta_\infty)}.$$

Τα μέτρα ρ, ν και $\frac{p}{r}$ έχουν ισοδύναμα αποτελέσματα.

# ΚΕΦΑΛΑΙΟ 3. ΥΠΟΛΟΓΙΣΜΟΣ ΒΕΛΤΙΣΤΗΣ ΤΡΙΓΩΝΟΠΟΙΗΣΗΣ

---



---

Ιδιαίτερη σημασία έχουν οι αλγόριθμοι χωρισμού σε τρίγωνα ενός συνόλου σημείων, οι οποίοι είναι βέλτιστοι (*optimal triangulation*) ως προς κάποιο κριτήριο και τέτοιο είναι το ζήτημα και αυτής της εργασίας.

Κριτήρια βελτιστοποίησης, για τον χωρισμό σε τρίγωνα, αποτελούν:

- το μήκος των ακμών (edge length)
- το μέγεθος των γωνιών (angle)
- το εμβαδόν (area) του τριγώνου.

### 3.1. Δυναμικός Προγραμματισμός

Πρόκειται για μια τεχνική σχεδίασης αλγορίθμου η οποία εφαρμόστηκε αρχικά στα Εφαρμοσμένα Μαθηματικά ως μέθοδος βελτιστοποίησης διαδικασιών απόφασης. Επιλύει ένα πρόβλημα αναλύοντάς το σε επιμέρους επικαλυπτόμενα υποπροβλήματα χρησιμοποιώντας την αναδρομή.

Συγκεκριμένα όλα τα επικαλυπτόμενα υποπροβλήματα λύνονται από την αρχή και τα αποτελέσματά τους καταγράφονται σε έναν πίνακα, από τον οποίο μπορούμε να βρούμε τη λύση για το αρχικό πρόβλημα.

Εφαρμόζεται κατά κανόνα σε προβλήματα βελτιστοποίησης, όπου αναζητείται σε ένα πλήθος διαφορετικών λύσεων η λύση που έχει τη βέλτιστη (μέγιστη ή ελάχιστη) τιμή.

Η δομή της βέλτιστης λύσης τίθεται κάθε φορά στην αρχή του αλγορίθμου. Στη συνέχεια ορίζεται αναδρομικά η τιμή της βέλτιστης λύσης, υπολογίζεται από κάτω προς τα πάνω ή το αντίστροφο, σε ορισμένες περιπτώσεις και κατασκευάζεται η βέλτιστη λύση από τα δεδομένα που έχουν υπολογιστεί.

Το πρώτο βήμα στην επίλυση ενός προβλήματος βελτιστοποίησης μέσω δυναμικού προγραμματισμού είναι ο χαρακτηρισμός της βέλτιστης λύσης. Ένα πρόβλημα εμφανίζει βέλτιστη υποδομή αν μια βέλτιστη λύση του εμπεριέχει βέλτιστες λύσεις σε υπό-προβλήματα. Η ύπαρξη βέλτιστης υποδομής σε ένα πρόβλημα αποτελεί ένδειξη ότι πιθανό να μπορεί να εφαρμοστεί ο δυναμικός προγραμματισμός.

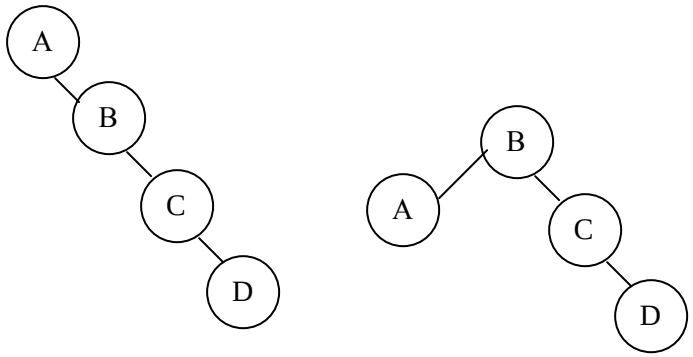
Στον δυναμικό προγραμματισμό κατασκευάζουμε μια βέλτιστη λύση στο πρόβλημα από βέλτιστες λύσεις σε υπό-προβλήματα (*αρχή του βέλτιστου-principle of optimality*).

Επομένως ένα πρόβλημα δυναμικού προγραμματισμού επιδεικνύει τα ακόλουθα γνωρίσματα:

- Διασπάται πάντα σε απλά υποπροβλήματα, ο ορισμός των οποίων δεν είναι μοναδικός και εξετάζεται κάθε φορά.
- Η βέλτιστη ολική λύση προκύπτει με συνδυασμό των βέλτιστων λύσεων των υποπροβλημάτων.
- Τα υποπροβλήματα μοιράζονται βέλτιστες λύσεις κοινών υποπροβλημάτων. Το πλήθος των υποπροβλημάτων που τελικά θα επιλυθούν πρέπει να είναι το πολύ πολυωνυμικό.

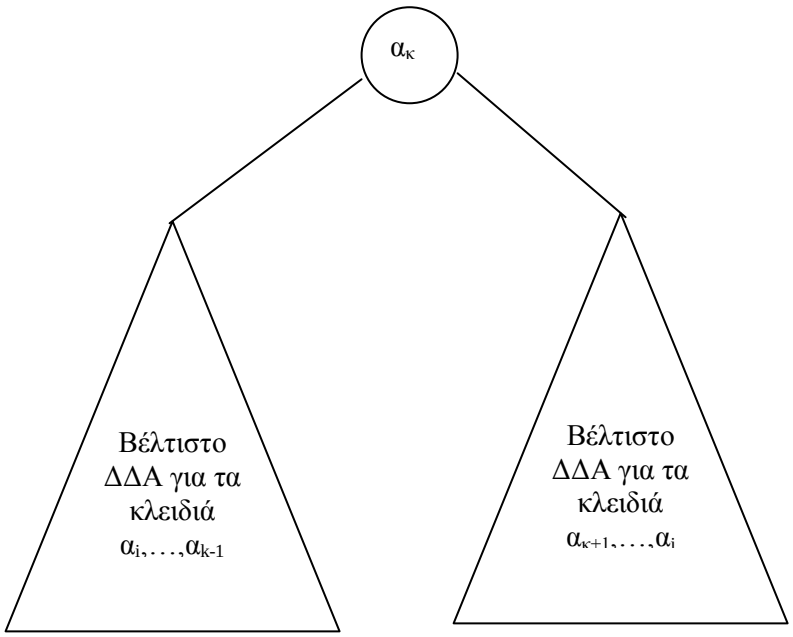
Παρακάτω θα αναφερθεί σαν παράδειγμα προβλήματος δυναμικού προγραμματισμού τα *βέλτιστα δυαδικά δένδρα αναζήτησης* (*Binary Search Trees*). Το ζητούμενο ενός τέτοιου δένδρου αναζήτησης είναι ο χρόνος αναζήτησης ενός στοιχείου στο δένδρο να είναι ο βέλτιστος δυνατός. Εάν υποτεθεί ότι σε κάθε στοιχείο αντιστοιχίζεται μια πιθανότητα αναζήτησης (π.χ. από επεξεργασία δεδομένων προηγούμενων αναζητήσεων), αναζητείται ο μέσος αριθμός συγκρίσεων μιας αναζήτησης να είναι ο ελάχιστος. Εδώ ισχύει επίσης ότι το πλήθος των διαφορετικών δένδρων αναζήτησης ισούται με τους αριθμούς *Catalan*, οπότε μια διεξοδική εύρεση όλων των συνδυασμών δυαδικών δένδρων για την εύρεση του καλύτερου ως προς μια συγκεκριμένη αναζήτηση ξεφεύγει κατά πολύ από την υλοποίηση αφού η συνάρτηση με την οποία προσεγγίζεται είναι η $4^n / n^{1.5}$ και προσεγγίζει το άπειρο ταχύτατα.

Ένα παράδειγμα δυαδικού δένδρου για τέσσερα κλειδιά (A, B, C και D) παρουσιάζεται σχηματικά παρακάτω.




Σχήμα 3-1 Δύο Πιθανά Δυαδικά Δένδρα Αναζήτησης Κλειδιών A, B, C και D

Έστω στη γενική περίπτωση υπάρχουν n κλειδιά $\alpha_1$, $\alpha_2$,…,$\alpha_n$ ταξινομημένα από το μικρότερο στο μεγαλύτερο και $p_1$, $p_2$,…,$p_n$ οι αντίστοιχες πιθανότητες αναζήτησης για καθένα από τα παραπάνω κλειδιά. Έστω επίσης, C[i,j] ο πίνακας που καταχωρείται ο μικρότερος μέσος αριθμός συγκρίσεων που πραγματοποιούνται σε μια επιτυχή αναζήτηση σε ένα δυαδικό δένδρο αναζήτησης $T_i^j$ , που αποτελείται από τα κλειδιά $a_i$,…,$a_j$, όπου i, j ακέραιοι δείκτες, με $1 \leq i$ και $j \leq n$. Για την λύση του προβλήματος αρκεί να βρεθεί η τιμή του C[1, n]. Εφαρμόζοντας την τεχνική του δυναμικού προγραμματισμού το πρόβλημα επιλύεται υπολογίζοντας τις τιμές των C[i, j]. Έστω $\alpha_\kappa$ η ρίζα ενός βέλτιστου δυαδικού δένδρου με αριστερό ($T_i^{\kappa-1}$ ) και δεξιό ($T_{\kappa+1}^{j+1}$ ) υποδένδρο σε βέλτιστη διευθέτηση. Απεικονίζεται σχηματικά παρακάτω.




Σχήμα 3-2 Δυαδικό Δένδρο Αναζήτησης με Ρίζα $\alpha_\kappa$

Η αναδρομική σχέση για τον υπολογισμό του C[i,j] είναι η ακόλουθη, όπου *depth*(v, T) είναι το βάθος (η απόσταση από τη ρίζα) του κόμβου v στο δέντρο T:

$$C[\mathrm{i},\mathrm{j}] = min_{\,1\le\kappa\le\mathrm{j}}\{p_\kappa + \sum_{s=i}^{k-1} p_s \cdot (depth(\alpha_s, \mathrm{T}_i^{k+1}) + 1) + \sum_{s=k+1}^{j} p_s \cdot (depth(\alpha_s, \mathrm{T}_{k+1}^{j}) + 1)\}$$

Και ύστερα από πράξεις καταλήγει στην αναδρομική σχέση για το C[i, j]:

$$C[i, j] = min_{\,1\le\kappa\le\mathrm{j}}\{C[i, \kappa\text{-}1] + C[\kappa\text{+}1, j], j\} + \sum_{s=i}^{j} p_s, \quad \forall 1 \le i \le n \qquad \text{Εξ. 3.1}$$

Ισχύει: $C[i, i\text{-}1] = 0$ για $1 \le i \le n$ και $C[\mathrm{i}, \mathrm{i}] = p_i$, $\forall 1 \le i \le n$.

Ο αλγόριθμος που περιγράφηκε χρειάζεται και ακόμη έναν δισδιάστατο πίνακα, προκειμένου να αποθηκεύσει το βέλτιστο δένδρο, όπου θα καταγράφεται η τιμή του *κ*, για την οποία ελαχιστοποιείται η σχέση (3.1). Ο νέος πίνακας θα έχει στις θέσεις της κύριας διαγωνίου *R*(i, i) την τιμή i, για $1 \le \mathrm{i} \le n$. Οι καταχωρήσεις στον πίνακα αντιστοιχούν στους δείκτες των ριζών των βέλτιστων υπό-δένδρων, οπότε και μπορεί να κατασκευαστεί το σύνολο του δένδρου των *n* κλειδιών.

### 3.2. Ο Γενικός Αλγόριθμος

Ο γενικός αλγόριθμος στην υλοποίηση που παρουσιάζεται παρακάτω για τη βέλτιστη διαίρεση σε τρίγωνα με βάση διάφορα κριτήρια στηρίζεται στην τεχνική του δυναμικού προγραμματισμού. Συγκεκριμένα, ο αλγόριθμος ελαχιστοποιεί ή μεγιστοποιεί μια ποσότητα κάθε φορά, που θα υπολογίζεται ανάλογα με την επιλογή του χρήστη.

Το πρόβλημα, που ζητείται να λυθεί είναι το πρόβλημα του χωρισμού ενός κυρτού πολυγώνου *n* σημείων σε τρίγωνα, έτσι ώστε στο σύνολο όλων των δυνατών χωρισμών του πολυγώνου σε τρίγωνα, ο επιθυμητός χωρισμός να αποτελείται από τα τρίγωνα, που έχουν τη μέγιστη ή την ελάχιστη αντίστοιχα τιμή του κριτηρίου.

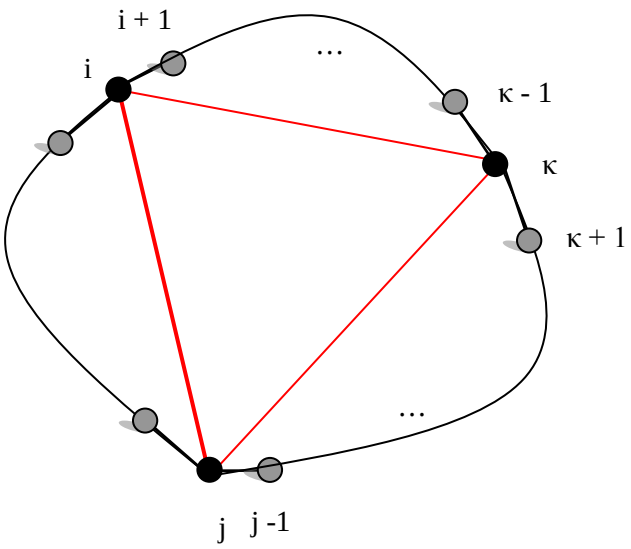
Τα κριτήρια που τίθενται για τη βελτιστοποίηση στην επιλογή ενός τριγώνου αφορούν τα εξής:

- το εμβαδόν του τριγώνου
- το μήκος της ακτίνας του περιγεγραμμένου κύκλου του τριγώνου, R
- το μήκος της ακτίνας του εγγεγραμμένου κύκλου του τριγώνου, r
- το λόγο της ακτίνας του περιγεγραμμένου προς την ακτίνα του εγγεγραμμένου κύκλου, ρ

- τη μικρότερη ή τη μεγαλύτερη γωνία ενός τριγώνου

Ένα υπό-πρόβλημα του γενικού προβλήματος ορίζεται να αφορά τη λύση μιας διαγωνίου ή μιας ακμής του πολυγώνου. Το πλήθος των διαγωνίων ενός πολυγώνου ισούται με τους συνδυασμούς ανά δύο των *n* κορυφών του, είναι δηλαδή της τάξης $O(n^2)$. Οπότε, μπορεί να εφαρμοστεί η τεχνική του δυναμικού προγραμματισμού, η οποία ζητάει πολυωνυμικό αριθμό υπό-προβλημάτων.

Ένα τυχαίο στιγμιότυπο του αλγορίθμου απεικονίζεται σχηματικά παρακάτω.




Σχήμα 3-3 Διάσπαση Γενικού Προβλήματος σε Υπό-προβλήματα με Βάση τη Διαγώνιο <i, j> στο Βήμα κ.

Η περιγραφή του αλγορίθμου του δυναμικού προγραμματισμού απεικονίζεται παρακάτω παρουσιάζοντας τα βασικά βήματα του αλγορίθμου καθώς και τη συνάρτηση *f*, που καλείται αναδρομικά, και τη *Λύση* κάθε υπό-προβλήματος τα οποία θα αναλυθούν λεπτομερέστερα.

Για *βήμα* = 2, …,*n*-1
  Για *i* = 0, …,*n*-1
    Για *k*= *i* +1, …,*i* + *βήμα* -1
      **Λύση** [*i*, *i* + *βήμα*] = **f (** **Λύση** [i, k], **criterion**(i , k, i+ βήμα), **Λύση** [k, i+ βήμα] **)**

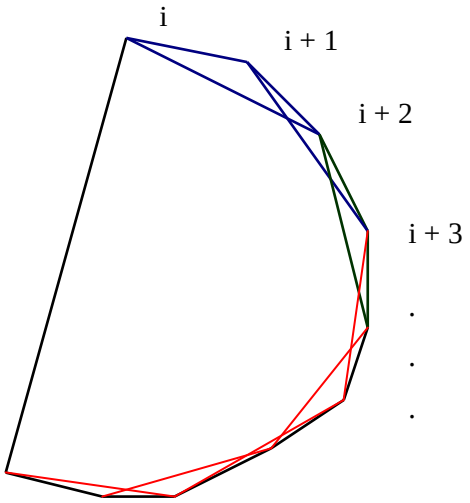
Σχήμα 3-4 Η Γενική Μορφή του Αλγόριθμου

Οι μεταβλητές *i, k* και *βήμα* αναπαριστούν τους δείκτες των σημείων, που θεωρούμε κάθε φορά και είναι δείκτες των κορυφών του πολυγώνου. Η μεταβλητή *βήμα* δηλώνει το μέγεθος του υπό-πολυγώνου του οποίου αναζητούμε τη βέλτιστη λύση κάθε φορά. Η μεταβλητή *i* αναπαριστά τη πρώτη κορυφή κάθε διαγωνίου και παίρνει όλες τις τιμές από 0 έως *n*- 1. Η μεταβλητή *k* διατρέχει όλες τις ενδιάμεσες κορυφές από τις *i* +1 έως την *i* + *βήμα* -1, συμπεριλαμβανόμενες και τις ίδιες, έτσι ώστε να εξετάσει τη σύνθεση της λύσης για το υπό-πρόβλημα (*i, i* + *βήμα*) με το βέλτιστο τρόπο από τις βέλτιστες λύσεις στα υπό-προβλήματα (*i, k*), (*k, i*+ *βήμα*) και της τιμής του κριτηρίου για το τρίγωνο **Δ** (*i, k, i*+ *βήμα* ). Η σύνθεση των λύσεων των υπό-προβλημάτων (*i, k*) και (*k, i*+ *βήμα*) και της τιμής του κριτηρίου του τριγώνου **Δ** (*i, k, i*+ *βήμα* ) πραγματοποιείται με τη συνάρτηση *f*.

Η συνάρτηση *f* θεωρεί εκείνη την τιμή του *k* η οποία βελτιστοποιεί (μεγιστοποιεί ή ελαχιστοποιεί) την τιμή του κριτηρίου, θεωρώντας ότι όλα τα ενδιάμεσα υπό-προβλήματα έχουν λυθεί και η τιμή τους έχει καταχωρηθεί στον δισδιάστατο πίνακα ***Λύση***[*i, j*] (***Αρχή του βέλτιστου του δυναμικού προγραμματισμού***).

Παρακάτω παρατίθενται σχηματικά για ένα τυχαίο πολύγωνο το στιγμιότυπο του προβλήματος για σταθερή τιμή της μεταβλητής *βήμα* ίσης με 2 και *k* ίσης με *i* +1 με την τιμή της *i* να κυμαίνεται από 0 έως *n* και ενός στιγμιότυπου με τυχαίες τιμές των *βήμα*, *i* και *k*.

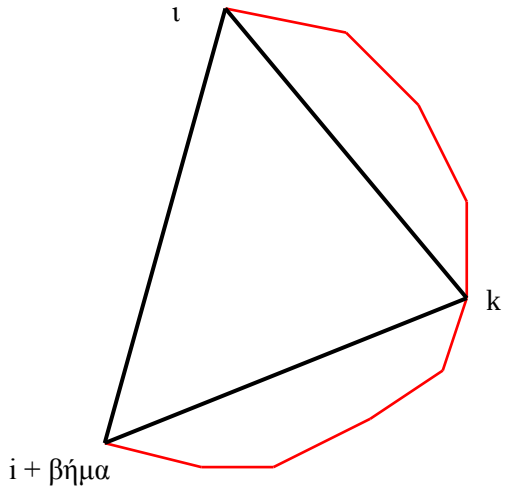
Στην πρώτη περίπτωση αντιστοιχίζεται η βέλτιστη τιμή για όλα τα υπό-προβλήματα (*i, i*+2) για όλα τα *i*: $0 \leq i \leq n$ και η οποία είναι ίση με τη τιμή του κριτηρίου για τα τρίγωνα **Δ** (*i* ,*i*+ 1, *i*+ 2), που όπως απεικονίζεται και σχηματικά παρακάτω, πρόκειται για όλα τα τρίγωνα, που σχηματίζονται με δύο διαδοχικές ακμές στο σύνορο του πολυγώνου και τη διαγώνιο, που τις συνδέει.




Σχήμα 3-5 Η Εύρεση της Τιμής του Κριτηρίου στα Τρίγωνα που Σχηματίζονται στο Σύνορο του Πολυγώνου

Στο σημείο αυτό αξίζει να σημειωθεί ότι η τιμή της συνάρτησης ***Λύση*** για τα υπό-προβλήματα των ακμών του πολυγώνου, της μορφής (*i*, *i*+1) ή <*i*, *i*-1> δεν ορίζεται, οπότε και η λύση στον αναδρομικό τύπο είναι η τιμή του κριτηρίου στα τρίγωνα **Δ**(*i*, *i*+1, *i*+2).

Στη γενική του μορφή το στιγμιότυπο παρουσιάζεται σχηματικά παρακάτω για τυχαίες τιμές των *i*, *βήμα* και *k*.




Σχήμα 3-6 Στιγμιότυπο του Προβλήματος για Τυχαίες Τιμές των *i*, *βήμα* και *κ*.

Η χρονική πολυπλοκότητα του αλγορίθμου είναι της τάξης του $O(n^3)$ και η χωρική πολυπλοκότητα της τάξης του $O(n^2)$ .

### 3.3. Βέλτιστοι Αλγόριθμοι

Τέτοιοι βέλτιστοι αλγόριθμοι που θεωρούν μια διαίρεση σε τρίγωνα, που μεγιστοποιεί ή ελαχιστοποιεί την τιμή ενός κριτηρίου του τριγώνου έχουν υπάρξει διάφοροι. Ο αλγόριθμος που ελαχιστοποιεί τη μέγιστη τιμή ενός κριτηρίου ονομάζεται Min_Max, ενώ εκείνος που μεγιστοποιεί την ελάχιστη τιμή του κριτηρίου ονομάζεται Max_Min.

Το πρόβλημα της αυτόματης παραγωγής βέλτιστων τριγωνοποιήσεων υπήρξε θέμα έρευνας από το 1960. Οι άπληστες (greedy) προσεγγίσεις (οι οποίες εξαλείφουν τα τρίγωνα από το «χειρότερο» προς

το «καλύτερο») αποκλείονται λόγω της NP-πληρότητας [8] του προβλήματος απόφασης, όπου αποφασίζει για ένα δοθέν σύνολο ακμών και ένα δοθέν σύνολο κορυφών εάν κάποιο υποσύνολο των ακμών ορίζει τριγωνοποίηση του συνόλου των κορυφών.

Τα περισσότερα θετικά αποτελέσματα επιφέρει η *Delaunay* διαίρεση σε τρίγωνα [7]. Έχει αποδειχθεί ότι στο σύνολο των δυνατών τριγωνοποιήσεων η *Delaunay* διαίρεση σε τρίγωνα μεγιστοποιεί την ελάχιστη γωνία τριγώνου, ελαχιστοποιεί την ακτίνα του μέγιστου περιγεγραμμένου κύκλου του τριγώνου και ελαχιστοποιεί την ακτίνα του μέγιστου πιο μικρού εγγεγραμμένου κύκλου ενός τριγώνου.

Οι *Edelsbrunner, Tan* και *Waupotitsch* σχεδίασαν αλγόριθμο πολυωνυμικού χρόνου, ο οποίος ελαχιστοποιεί τη μέγιστη γωνία τριγώνου [6]. Ο αλγόριθμος αυτός κατασκευάζει μια min_max angle τριγωνοποίηση εισάγοντας επαναληπτικά μια νέα ακμή, αφαιρώντας τις παλιές ακμές, οι οποίες τέμνονται από την καινούρια, ξανά-τριγωνοποιώντας τις τρύπες του πολυγώνου και από τα δύο μέρη της νέας ακμής. Επίσης, οι *H. Edelsbrunner* και *T.S. Tan* υπολογίζουν έναν τετραγωνικό αλγόριθμο για την εύρεση της min_max length τριγωνοποίησης ενός πολυγώνου [4].

Οι *M. Bern, H. Edelsbrunner, D. Eppstein, S. Mitchell* και *T.S. Tan* [8] στηρίζονται στην τεχνική της εισαγωγής ακμής (*edge insertion*) και υπολογίζουν βέλτιστες τριγωνοποιήσεις σε πολυωνυμικό χρόνο. Συγκεκριμένα, υπολογίζουν έναν $O(n^2 \log n)$ αλγόριθμο που μεγιστοποιεί το ελάχιστο ύψος τριγώνου (*max_min height*), έναν $O(n^3)$ αλγόριθμο που ελαχιστοποιεί τη μέγιστη τιμή της εκκεντρικότητας (*η απόσταση από το κέντρο του περιγεγραμμένου κύκλου*) ενός τριγώνου (*min_max eccentricity*) και έναν $O(n^3)$ αλγόριθμο, ο οποίος υπολογίζει μια τριγωνοποιημένη επιφάνεια, που παρεμβάλλει σημεία στον $\Re^3$, με min_max gradient (*κλίση*).

Επίσης, ένας γραμμικός αλγόριθμος, ο οποίος μεγιστοποιεί το ελάχιστο μήκος ακμής τριγώνου ενός κυρτού πολυγώνου (*Max_Min length triangulation of a convex polygon*).

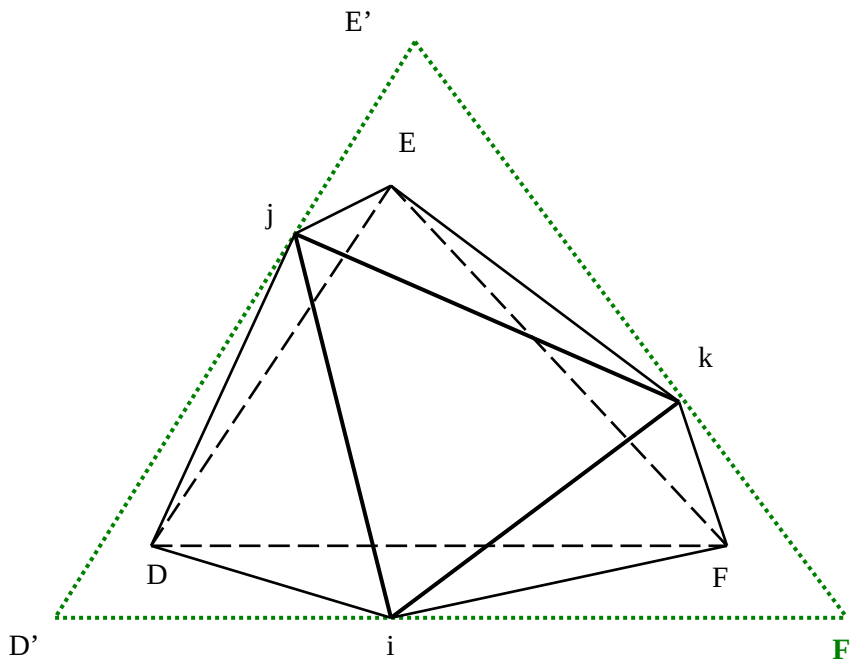
Ο αλγόριθμος των *Keil* & *Vassilev* υπολογίζει την Max_Min και την Min_Max διαίρεση σε τρίγωνα ενός πολυγώνου, ως προς το κριτήριο του εμβαδού του τριγώνου. Επίσης υπάρχει ο αλγόριθμος *Minimum Weight Triangulation*, ο οποίος ελαχιστοποιεί το άθροισμα των μηκών των πλευρών του συνόλου των τριγώνων.

*3.3.1. Ο Αλγόριθμος των Keil & Vassilev*

Ο αλγόριθμος των *Keil & Vassilev* υπολογίζει τη βέλτιστη τριγωνοποίηση ενός κυρτού πολυγώνου σε τρίγωνα μεγιστοποιώντας το εμβαδόν του ελάχιστου τριγώνου στη μια περίπτωση και ελαχιστοποιώντας το εμβαδόν του μεγαλύτερου τριγώνου στη δεύτερη περίπτωση στο σύνολο των δυνατών διαιρέσεων σε τρίγωνα του πολυγώνου. Ο αλγόριθμός τους στηρίζεται στο δυναμικό προγραμματισμό και εφαρμόζοντας κατάλληλες γεωμετρικές δομές καταφέρνουν πολυπλοκότητα χρόνου της τάξης $O(n^2 \log n)$ και πολυπλοκότητα χώρου $O(n^2)$ [6].

Η συγκεκριμένη εργασία στηρίζεται σε ένα αριθμό γεωμετρικών ιδιοτήτων των κυρτών πολυγώνων και των τριγώνων, βάσει των οποίων αποδεικνύονται και οι διάφοροι ισχυρισμοί, που οδηγούν στην επίλυση του προβλήματος στο ζητούμενο χρόνο. Μεταξύ άλλων αποδεικνύονται τα εξής:

Δοθέντος ενός κυρτού πολυγώνου P στο επίπεδο και μιας τριγωνοποίησης T του P, το τρίγωνο με το μικρότερο εμβαδόν έχει μια τουλάχιστον ακμή στο σύνορο του P [6] (Λήμμα 3).




Σχήμα 3-7 Το Χειρότερο Τρίγωνο

Το παραπάνω στηρίζεται στην εξής γεωμετρική ιδιότητα:

Δοθέντος ενός τριγώνου **Δ***DEF* στο επίπεδο και ενός τριγώνου **Δ***PQR* εγγεγραμμένου σε αυτό, τέτοιο ώστε $P \in EF$, $Q \in FD$, $R \in DE$: $\mathbf{A}_{\Delta PQR} \geq \min(\mathbf{A}_{\Delta DQR}, \mathbf{A}_{\Delta ERP}, \mathbf{A}_{\Delta FPQ})$, όπου **A** το εμβαδόν του

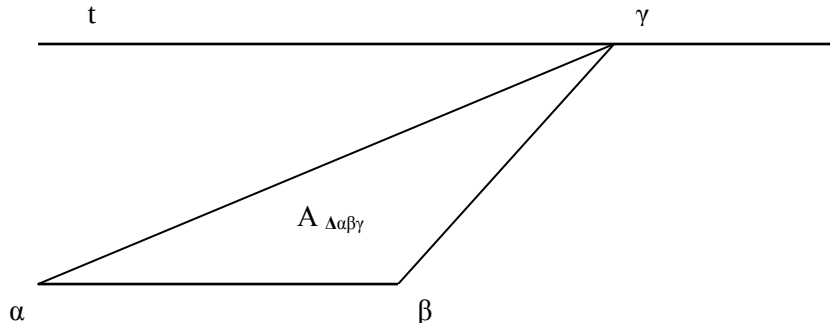
αντίστοιχου τριγώνου. Με άλλα λόγια, διατυπώνεται ότι το μικρότερο τρίγωνο δεν είναι απαραίτητα εκείνο που είναι εσωτερικά σε κάποιο άλλο.

Μια άλλη ιδιότητα που αφορά τα κυρτά πολύγωνα είναι και η εξής:

Η απόσταση ανάμεσα στην ευθεία που διέρχεται από μια ακμή ενός κυρτού πολυγώνου και στις κορυφές του πολυγώνου, όπως εκείνες συναντώνται κατά τη φορά των δεικτών του ρολογιού (ή αντίστροφα προς τη φορά των δεικτών του ρολογιού) έχει ένα μόνο μέγιστο (*unimodal*).

Η παραπάνω ιδιότητα επεκτείνεται και στο εμβαδόν των τριγώνων, που σχηματίζονται από μια βάση ακμή σε ένα κυρτό πολύγωνο, στο οποίο παρατηρείται επίσης ένα μέγιστο.

Διαφορετικά, θεωρείται μια τιμή κατωφλίου, τ, η οποία ορίζεται από μια ευθεία παράλληλη προς την ακμή βάσης ενός τριγώνου, όπως απεικονίζεται παρακάτω στο σχήμα, έτσι ώστε κάθε σημείο πάνω στην ευθεία σχηματίζει ένα τρίγωνο (με τα άκρα της ακμής βάσης) εμβαδού ίσου με τ. Παρόμοια, εάν η συνάρτηση έχει ένα μέγιστο, τότε η τομή της *γραμμής κατωφλίου* (*threshold line*) με το εσωτερικό του πολυγώνου είναι ένα μοναδικό τμήμα (ή διαφορετικά η γραμμή με το πολύγωνο δεν έχουν κανένα σημείο τομής).




Σχήμα 3-8 Ευθεία Κατωφλίου t του Τμήματος αβ για Εμβαδόν τ

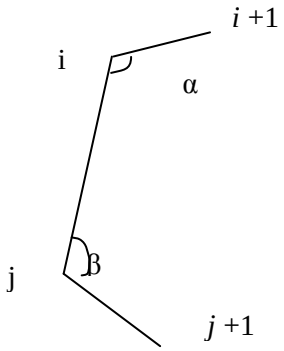
Είναι χρήσιμο στο σημείο αυτό να αναφερθούν κάποιοι ορισμοί για την κατανόηση της ανάλυσης που ακολουθεί. Για ένα κυρτό πολύγωνο ορίζονται τα εξής:

Η κορυφή *Top*(*i, j*) του ζεύγους κορυφών (*i, j*) του P δηλώνει την πιο μακρινή κορυφή του P στο διάστημα [*i, j*] από την ευθεία τη διερχόμενη από την ακμή *i j* . Όταν είναι δύο οι κορυφές με την ίδια απόσταση θεωρείται εκείνη που προηγείται κατά την ωρολογιακή διάσχιση των κορυφών από τη i

στην *Top*(*i, j*). Ο υπολογισμός των τιμών της *Top* για το σύνολο των ακμών και των διαγώνιων του P πραγματοποιείται σε $O(n^2)$ χρόνο.

Η ζωνικότητα (*zonality*) ενός υπό-πολυγώνου $P_{ij}$ ορίζεται ως εξής: ένα πολύγωνο $P_{ij}$ έχει τιμή ζωνικότητας $\kappa \in \{1, 2, 3, 4\}$, εάν και μόνο εάν
$\kappa = \angle j\ i\ (i+1) + \angle\ (j-1)\ j\ i\ ) / 90^0$.

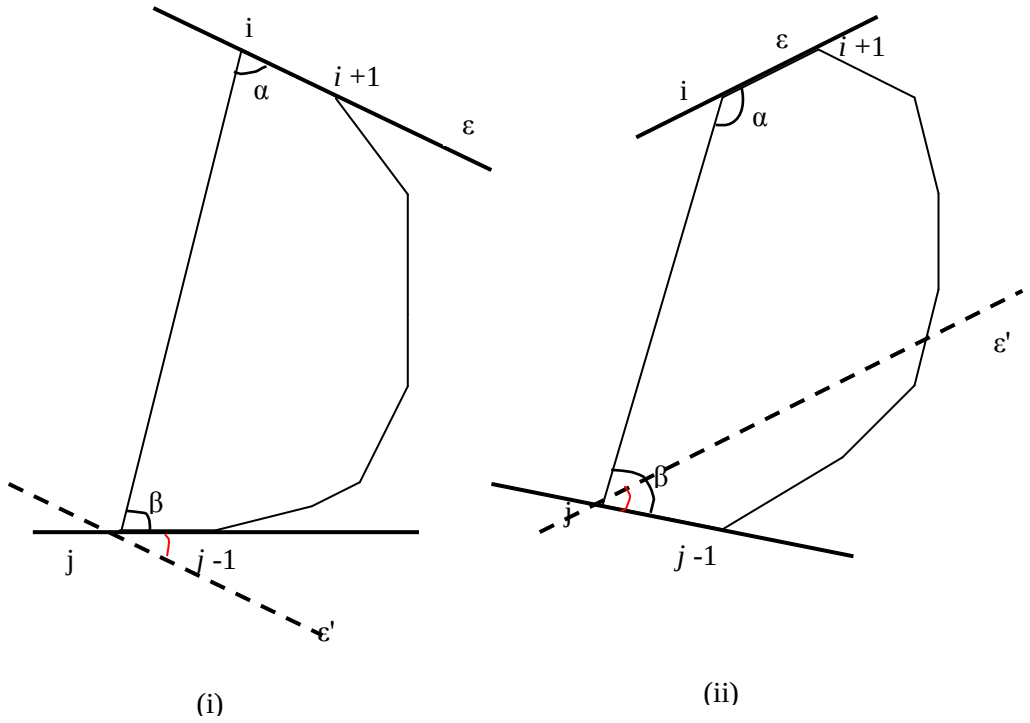
Εάν συμβολιστούν οι γωνίες $\angle j\ i\ (i+1)$, $\angle\ (j-1)\ j\ i$ , ως $\alpha$ και $\beta$ αντίστοιχα, τότε
$\kappa = (\alpha + \beta) / 90^0$.




Σχήμα 3-9 Zonality υπό-πολυγώνου

Ένα υπό-πολύγωνο $P_{ij}$ έχει ζωνικότητα ίση με δύο, όταν όλες οι κορυφές του υπό-πολυγώνου ανάμεσα στις κορυφές *i* και *j* τοποθετούνται ανάμεσα σε δύο παράλληλες λωρίδες (*parallel strips*) που διέρχονται από τα άκρα της ακμής-βάσης <*i, j*>.

Ένα 2-*zone* και 3-*zone* υπό-πολύγωνο $P_{ij}$ απεικονίζονται σχηματικά παρακάτω.




Σχήμα 3-10 Δύο υπό-πολύγωνα $\boldsymbol{P_{ij}}$ : 2-*zone* και 3-*zone* αντίστοιχα

Η ευθεία ε εφάπτεται στην ακμή (*i*, *i*+1) και η ευθεία ε' διέρχεται από το σημείο j και είναι παράλληλη στην ε. Οπότε στο σχήμα 2.9 (i) διακρίνεται ότι το υπό-πολύγωνο $P_{ij}$ βρίσκεται ανάμεσα στις παράλληλες λωρίδες, που ορίζονται από τις ευθείες ε και ε' αντίστοιχα.και το άθροισμα των γωνιών *α* και *β* είναι $\alpha + \beta \leq 180^0$, οπότε η *zonality* του υπό-πολυγώνου $P_{ij}$ είναι ίση με δύο (κ =2).

Ενώ στο σχήμα 2.9 (ii) ένα κομμάτι του υπό-πολυγώνου βρίσκεται εκτός των παράλληλων λωρίδων ε και ε', το άθροισμα των προσκείμενων γωνιών στη ij διαγώνιο είναι: $180^0 \leq \alpha + \beta \leq 270^0$, οπότε η τιμή της *zonality* για το υπό-πολύγωνο $P_{ij}$ είναι ίση με τρία (κ =3).

Η συνάρτηση, που αναπαριστά τη *zonality* ενός υπό-πολυγώνου ορίζεται ως z(i, j) και είναι ίση με την τιμή *α*, εάν και μόνο εάν το $P_{ij}$ είναι *α-zone* υπό-πολύγωνο.

Για την *zonality* ενός πολυγώνου ισχύουν οι ισότητες:

$z(i, j) + z(j, i) \leq 5$ και

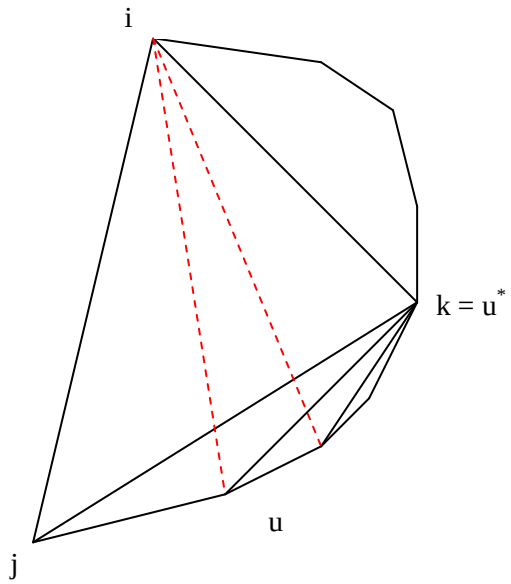
εάν *i, j* και *k* τρεις κορυφές σε ωρολογιακή φορά, τότε:

$z(i, j) + z(j, k) + z(k, i) \leq 6.$

Οι παραπάνω ιδιότητες είναι ιδιαιτέρως χρήσιμες στα επόμενα που θα ακολουθήσουν και αφορούν στον υπολογισμό του Max_Min εμβαδού της τριγωνοποίησης T για το κυρτό πολύγωνο P.

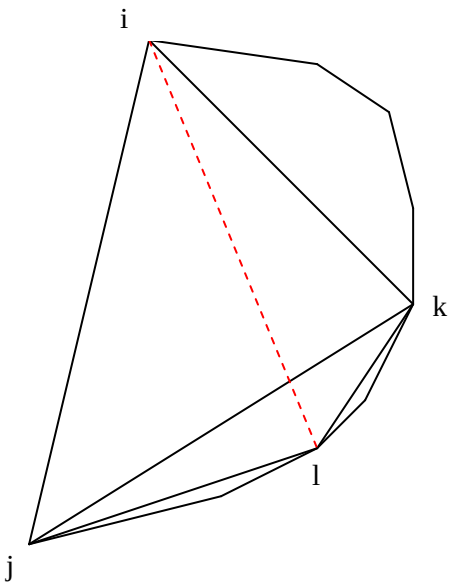
Στα επόμενα ως $\mu$(T) συμβολίζεται το εμβαδόν του ελάχιστου τριγώνου και ως $\lambda$(T) αντίστοιχα το εμβαδόν του μεγαλύτερου τριγώνου και αποτελούν τα μέτρα ποιότητας για τους αλγόριθμους βελτιστοποίησης Max_Min και Min_Max αντίστοιχα. Ως $M_1(P)$ συμβολίζεται η Max_Min τριγωνοποίηση και $\mu^*(P)$ το εμβαδόν του «χειρότερου» (πιο μικρού) τριγώνου. Αντίστοιχα, ως $M_2(P)$ συμβολίζεται η Min_Max τριγωνοποίηση και ως $\lambda^*(P)$ το εμβαδόν του «χειρότερου» (πιο μεγάλου) τριγώνου.

Το επόμενο λήμμα αποτελεί τη βάση για τον χωρισμό σε τρίγωνα του πολυγώνου P. Έστω $P_{ij}$ ένα 2-zone υπό-πολύγωνο. Δοθείσης μιας τιμής κατωφλίου, τ, εάν υπάρχει τριγωνοποίηση T του $P_{ij}$ τέτοια ώστε μ(T) ≥ τ, τότε υπάρχει τριγωνοποίηση T' του $P_{ij}$ τέτοια ώστε μ(T') ≥ τ, και η τριγωνοποίηση T' περιέχει ένα από τα τρίγωνα **Δ** i(i+1)j ή **Δ** i(j-1)j (Λήμμα 11) [6].
Η θεώρηση του παραπάνω λήμματος αποδίδεται σχηματικά στο παρακάτω σχήμα.




Σχήμα 3-11 Επανατριγωνοποίηση σε ένα 2-zone Πολύγωνο, Max_Min Εμβαδόν

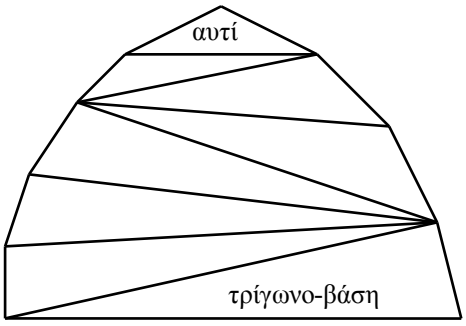
Για την Min_Max τριγωνοποίηση ισχύει ένα αντίστοιχο λήμμα, που είναι το εξής:
Για δεδομένη τιμή κατωφλίου, τ, εάν υπάρχει τριγωνοποίηση T, τέτοια ώστε λ(T)≤ τ, τότε υπάρχει νέα τριγωνοποίηση T' που περιέχει ένα από τα τρίγωνα **Δ** i(i+1)j ή **Δ**i(j-1)j.




Σχήμα 3-12 Επανατριγωνοποίηση σε ένα 2-zone Πολύγωνο, Min_Max Εμβαδόν

Η απεικόνιση της βέλτιστης τριγωνοποίησης εμβαδού φαίνεται παρακάτω. Έτσι, σε αυτή τη βέλτιστη τριγωνοποίηση περιέχονται τρίγωνα της μορφής **Δ** i(i+1)j ή **Δ** i(j-)j, γειτονικά σε μια ακμή-βάση *ij*.

Η βέλτιστη τριγωνοποίηση περιέχει ακριβώς δύο τρίγωνα «αυτιά» (το τρίγωνο βάσης και κάποιο ακόμα) και όλα τα υπόλοιπα τρίγωνα είναι συνοριακά (περιέχουν ακριβώς μια ακμή συνόρου). Δεν υπάρχουν εσωτερικά τρίγωνα, δηλαδή τρίγωνα των οποίων οι πλευρές είναι όλες διαγώνιοι του πολυγώνου.




Σχήμα 3-13 Δομή της Βέλτιστης Τριγωνοποίησης Εμβαδού για έως 2-zone πολύγωνα

Τέτοιες τριγωνοποιήσεις καλούνται *sleeves*. Στη γενική περίπτωση η βέλτιστη τριγωνοποίηση μπορεί να περιέχει το πολύ τρία «αυτιά», ένα εσωτερικό τρίγωνο και όλα τα υπόλοιπα να είναι συνοριακά.

Εάν *i* μια κορυφή του P, δηλώνονται οι κορυφές *MaxCW(i)* να είναι η τελευταία, σε ωρολογιακή φορά από την *i*, κορυφή του P για την οποία ισχύει $\mathbf{z}(i, MaxCW(i)) \leq 2$. Δηλαδή, όλα τα υπό-πολύγωνα στη σειρά $P_{i,i+1}$, $P_{i,i+2}$, ..., $P_{i,MaxCW(i)}$ έχουν *zonality* 2 ή μικρότερη και $\mathbf{z}(i, MaxCW(i)+1) \geq 3$. Ομοίως, ορίζεται η κορυφή *MaxCCW(i)* να είναι η τελευταία κατά την ανθωρολογιακή φορά διάσχισης από την κορυφή *i*, τέτοια ώστε $\mathbf{z}(MaxCCW(i), i) \leq 2$.

Για τις κορυφές *MaxCW(i)* και *MaxCCW(i)* ισχύουν οι ακόλουθες ιδιότητες:

$MaxCCW(i) = Top(i-1, i)$ και

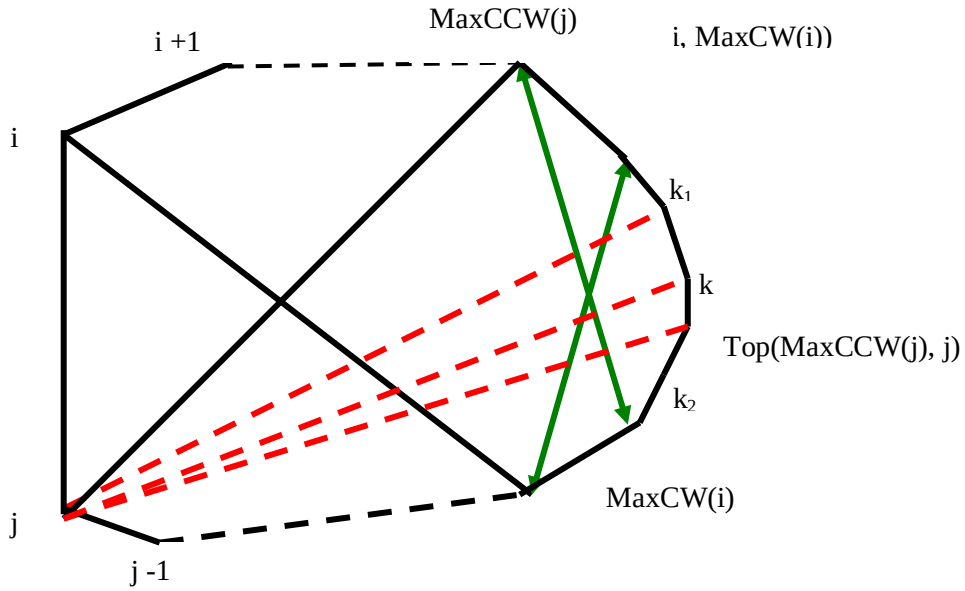
$MaxCW(i) = Top(i, i+1)$ ή

$MaxCW(i) = Top(i, i+1)+1$

Ακολουθούν δύο ακόμα λήμματα τα οποία καθορίζουν τα *διαστήματα παραδεκτότητας* (*admissibility intervals*) ως προς τις τιμές των *i* και τ .

Για τον αλγόριθμο *Max_Min*, έστω $P_{ij}$ ένα υπό-πολύγωνο του P και έστω το διάστημα των κορυφών από την *MaxCCW(j)* έως την *Top(MaxCCW(j), j).* Δοθείσης τιμής κατωφλίου τ, εάν υπάρχουν δύο κορυφές $\kappa_1$ και $\kappa_2$ στο παραπάνω διάστημα, τέτοιες ώστε υπάρχουν τριγωνοποιήσεις $T_1$ του $P_{\kappa 1j}$ και $T_2$ του $P_{\kappa 2j}$ για τις οποίες ισχύουν $\mu(T_1) \geq \tau$ και $\mu(T_2) \geq \tau$, τότε για κάθε κορυφή κ στο διάστημα $[\kappa_1, \kappa_2]$ υπάρχει τριγωνοποίηση T του $P_{kj}$ τέτοια ώστε $\mu(T) \geq \tau$. Το μεγαλύτερο τέτοιο διάστημα για όλα τα *i* καλείται διάστημα παραδεκτότητας ως προς τα *j* και *τ.*

Αποδίδεται στο παρακάτω σχήμα.

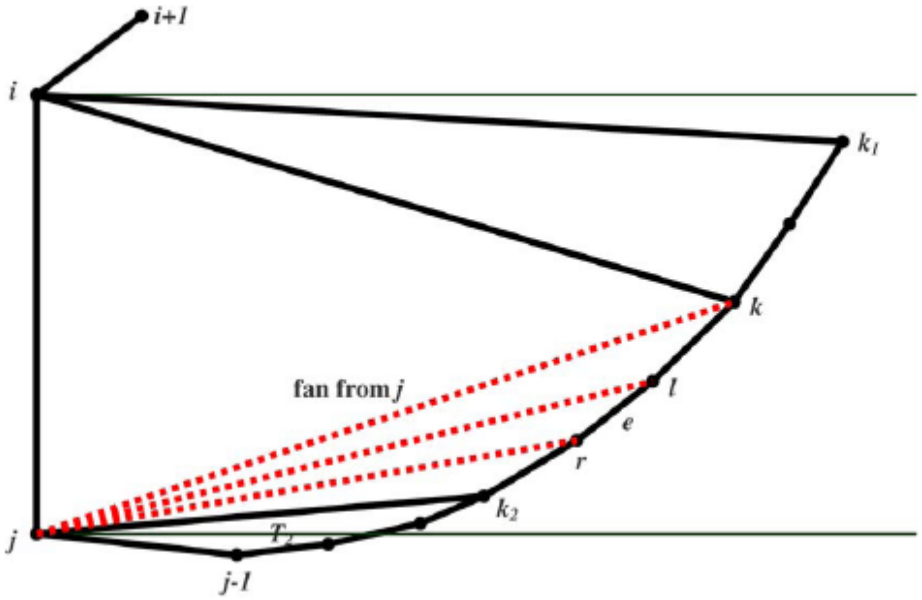



Σχήμα 3-14 Διαστήματα Παραδεκτότητας για τον Max_Min Χωρισμό.

Ισχύει ότι η συνάρτηση $\mu^*(P_{ik})$, της τιμής του μικρότερου εμβαδού για τον Max_Min χωρισμό έχει ένα μέγιστο στο διάστημα, όπου $k \in$ [Top(i, MaxCW(i)), MaxCW(i)], όπως επίσης ότι η συνάρτηση $\mu^*(P_{mi})$, έχει ένα μέγιστο για m ∈ [MaxCCW(i), Top(MaxCCW(i), i)].

Όσον αφορά τον Min_Max χωρισμό ισχύουν τα ακόλουθα:

Θεωρείται το σύνολο των κορυφών που έπονται της κορυφής Top(i,j) σε ωρολογιακή φορά και βρίσκονται στην παράλληλη λωρίδα των ευθειών που είναι κάθετες στα άκρα της ακμής βάσης ij στα ακραία σημεία της.




Σχήμα 3-15 Διαστήματα Παραδεκτότητας για τον *Min_Max* Χωρισμό.

Δοθείσης μιας τιμής κατωφλίου τ, εάν υπάρχουν δύο κορυφές $\kappa_1$ και $\kappa_2$ στο παραπάνω διάστημα τέτοιες ώστε υπάρχουν τριγωνοποιήσεις $T_1$ του $P_{k1j}$ και $T_2$ του $P_{k2j}$, για τις οποίες ισχύει ότι $\lambda(T_1) \leq \tau$, $E_{\Delta ik1j} \leq \tau$ και $\lambda(T_2) \leq \tau$, $E_{\Delta i2j} \leq \tau$, τότε για κάθε κορυφή κ μεταξύ των $\kappa_1$ και $\kappa_2$ υπάρχει τριγωνοποίηση T του $P_{k1j}$ τέτοια ώστε $\lambda(T) \leq \tau$, $E_{\Delta ikj} \leq \tau$. Το μεγαλύτερο τέτοιο διάστημα, ως προς j και τ, καλείται διάστημα παραδεκτότητας για τον Min_Max αλγόριθμο.

Η συνάρτηση $\lambda^*(P_{ik})$ έχει ένα μέγιστο στο διάστημα, όπου $\kappa \in [i,Top(i, j)]$ και η $\lambda^*(P_{mi})$ έχει ένα μέγιστο στο διάστημα, όπου $m \in [Top(j,i),i]$ ευρισκόμενο στη περιοχή, όπου σχηματίζεται από τις δύο παράλληλες ευθείες, κάθετες στα άκρα της ij, για κάθε κορυφή j.

Περιγραφικά μια γενική μορφή του αλγορίθμου δίνεται παρακάτω.

**Έξοδος**: Τριγωνοποίηση $\mathbf{T_1}$ (τριγωνοποίηση *MaxMin* εμβαδού ) του P τέτοια ώστε το τρίγωνο ελάχιστου εμβαδού στην $\mathbf{T_1}$ έχει το μεγαλύτερο εμβαδόν στο σύνολο όλων των δυνατών τριγωνοποιήσεων του **P** και Τριγωνοποίηση $\mathbf{T_2}$ (τριγωνοποίηση *MinMax* εμβαδού) τέτοια ώστε το τρίγωνο με το μεγαλύτερο εμβαδόν στην $\mathbf{T_2}$ έχει το μικρότερο εμβαδόν στο σύνολο των τριγωνοποιήσεων του **P**.

(i) Υπολόγισε για κάθε ακμή και διαγώνιο του **P**, την πιο απομακρυσμένη κορυφή.
Καταχώρησε τα δεδομένα στον πίνακα **Top**[0..n-1, 0..n-1].
Υπολόγισε τις τιμές των πινάκων **MaxCW**[0..n-1] και **MaxCCW** [0..n-1] από τις τιμές του πίνακα **Top**[0..n-1,0..n-1].

(ii) Αρχικοποίησε τις τιμές του πίνακα SubPr[0..n-1,0..n-1] με -1.
Για i = 0 έως n -1
Θέσε τις τιμές του πεδίου **εμβαδόν** του SubPr[i, (i +2) mod n] με την τιμή του εμβαδού στο τρίγωνο **Δi**(i+1)(i+2)
Θέσε τις τιμές του πεδίου **δείκτης** του SubPr[i, (i +2) mod n] με την τιμή (i+1) mod n.

(iii) Για l = 3 έως n-1
Για i = 0 έως n-1
Εάν z(i, i+l) ≤ 2 τότε
Θέσε τις τιμές του πεδίου **εμβαδόν** του SubPr[i, (i +l) mod n] με την ελάχιστη/μέγιστη τιμή του εμβαδού στη βέλτιστη λύση του $P_{i,i+l}$
Θέσε τις τιμές του πεδίου **δείκτης** του SubPr[i, (i +l) mod n] με το δείκτη της κορυφής που συνδέεται με την ακμή i(i+l) σε κάθε βέλτιστη τριγωνοποίηση του $P_{i(i+l)}$.

(iv) Για i = 3 έως n-1
Για j = 0 έως n-1
Εάν SubPr[j,i] ≠ -1
Εάν SubPr[i,j] ≠ -1
Σύγκρινε με την έως τώρα βέλτιστη λύση και ενημέρωσε αν χρειαστεί
Διαφορετικά
Αναζήτησε στο διάστημα [i,j] την κορυφή κ, για την οποία το τρίγωνο **Δ**ikj έχει τη λύση.
Σύγκρινε με τα έως τώρα αποτελέσματα και ενημέρωσε αν χρειαστεί.

(v) Κατασκεύασε τις δύο βέλτιστες Τριγωνοποιήσεις $T_1$ και $T_2$ που έχουν προκύψει μετά την εκτέλεση του βήματος (iv).

*Ο Αλγόριθμος των Keil & Vassilev περιγραφικά*

Όσον αφορά τον *Max_Min* αλγόριθμο για την επίτευξη χρονικής πολυπλοκότητας $O(n^2 \log n)$ χρησιμοποιούνται ισοσταθμισμένα δυαδικά δένδρα αναζήτησης για την αποθήκευση των τιμών των «χειρότερων» τριγώνων για το $P_{ik}$, για $k \in [i+1, MaxCW(i)]$ και για το $P_{ki}$, για $k \in [MaxCCW(i), i-1]$.

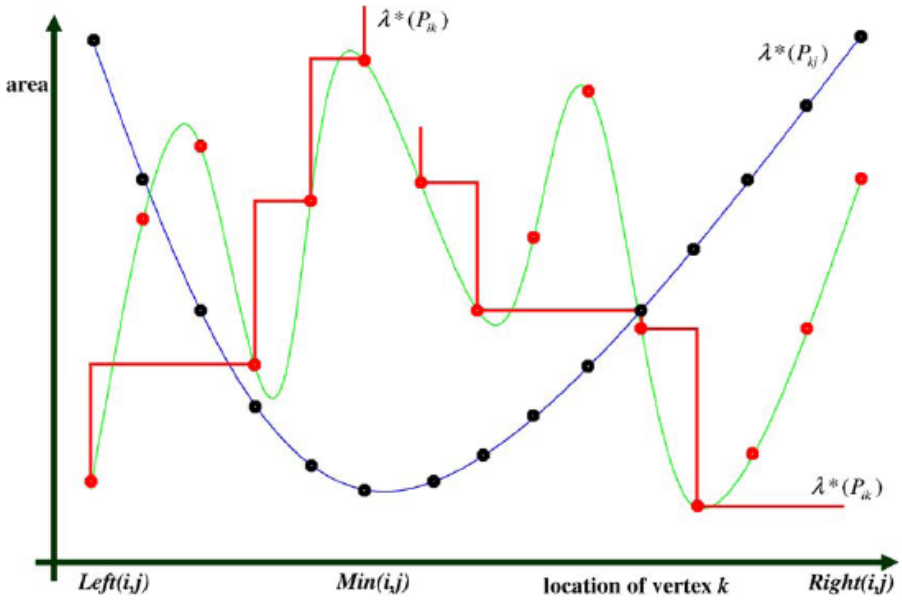
Οι τιμές των παραπάνω συναρτήσεων τοποθετούνται αρχικά σε έναν πίνακα σε ταξινομημένη διάταξη και υπολογίζονται για κάθε κορυφή του αρχικού πολυγώνου P.

Ο χρόνος αναζήτησης σε ισοσταθμισμένο δυαδικό δένδρο αναζήτησης είναι της τάξης $O(n \log n)$. Ο συνολικός χρόνος προεπεξεργασίας ανέρχεται στην τάξη $O(n^2 \log n)$ και ο χρόνος επεξεργασίας στον βρόγχο του βήματος (*iv*) του αλγορίθμου για κάθε διαγώνιο είναι της τάξης $O(\log n)$. Ο χώρος που απαιτείται είναι της τάξης $O(n^2)$.

Για τον *Min_Max* αλγόριθμο προκύπτει η ίδια χρονική και χωρική πολυπλοκότητα με τον *Max_Min* αλγόριθμο. Συγκεκριμένα, για κάθε διαγώνιο παρατηρείται *unimodality* για την περιοχή των κορυφών στο διάστημα των δύο παράλληλων λωρίδων κάθετων στα άκρα της διαγωνίου *ij*. Τα διαστήματα *unimodality* των συναρτήσεων $\lambda^*(P_{ik})$ και $\lambda^*(P_{kj})$ καλύπτουν το διάστημα ανάμεσα στις παράλληλες λωρίδες. Η κορυφή απέναντι στην μεγαλύτερη ακμή ενός τριγώνου βρίσκεται ανάμεσα από τις παράλληλες λωρίδες κάθετες στα άκρα της δεδομένης ακμής. Το γεγονός αυτό είναι αρκετό για να θεωρηθεί ότι η *Min_Max* τριγωνοποίηση περιέχει ένα τρίγωνο 2-2-2 *zone* το οποίο θα βρεθεί από τη μεγαλύτερη ακμή του, έστω *ij*, και τη τρίτη κορυφή με την οποία συνδέεται, έστω, *k* η οποία και βρίσκεται στην λωρίδα των παράλληλων ευθειών. Διαφορετικά, υπάρχει μια διαγώνιος η οποία έχει και τα δύο υπό-προβλήματα του δεξιού και του αριστερού υπό-πολυγώνου λυμένα.

Αρκεί να αναλυθεί η εύρεση της κορυφής κ στο διάστημα που καλύπτεται από τη λωρίδα των παράλληλων ευθειών και η οποία ελαχιστοποιεί τη μέγιστη τιμή των $\lambda^*(P_{ik})$ και $\lambda^*(P_{kj})$.

Ορίζονται η πιο αριστερή και η πιο δεξιά κορυφή της λωρίδας των ευθειών κάθετων στα άκρα της διαγωνίου *ij* να είναι αντίστοιχα η *Left*(i,j) και *Right*(i,j). Η κορυφή *Top*(i,j) βρίσκεται είτε στο διάστημα [*Left*(i,j), *Right*(i,j)], όπου σε αυτή την περίπτωση η $\lambda^*(P_{ik})$ είναι *unimodal* στο διάστημα [*Left*(i,j), *Top*(i,j)] και η $\lambda^*(P_{kj})$ στο διάστημα [*Top*(i,j), *Right*(i,j)], είτε βρίσκεται αριστερά της *Left*(i,j), όπου σε αυτή την περίπτωση μόνο η $\lambda^*(P_{kj})$ είναι *unimodal* στο διάστημα [*Left*(i,j),*Right*(i,j)], είτε τέλος ακολουθεί την *Right*(i,j) και σε αυτή την περίπτωση μόνο η $\lambda^*(P_{ik})$ είναι *unimodal* στο διάστημα [*Left*(i,j),*Right*(i,j)]. Αναλύεται η δεύτερη περίπτωση και έστω *Min*(i,j) η θέση της *k* με την ελάχιστη τιμή της $\lambda^*(P_{kj})$.



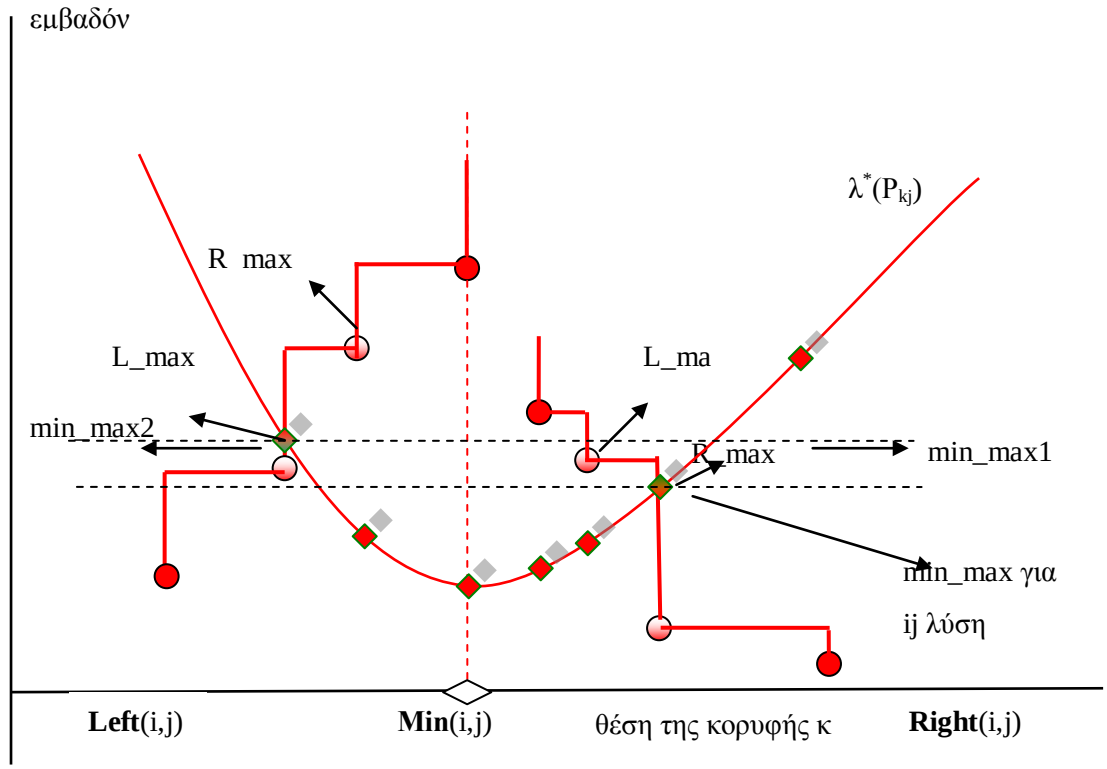

Σχήμα 3-16 Ανάλυση των Δομών για την Εύρεση της Κορυφής k με τη Βέλτιστη Λύση

Ενώ η $\lambda^*(P_{kj})$ παρουσιάζει ολικό ελάχιστο και η $\lambda^*(P_{ik})$ έχει αυξομειώσεις. Στο τμήμα δεξιά της κορυφής *Min*(i,j), όπου η $\lambda^*(P_{kj})$ αυξάνεται θεωρούνται τα σημεία της $\lambda^*(P_{ik})$ που μειώνουν το εμβαδόν. Τα δεδομένα σημεία παρουσιάζουν μια δομή σκάλας, όπου σημεία (k, $\lambda^*(P_{ik})$) απορρίπτονται όταν βρίσκονται βορειοανατολικά (ΒΑ) σε σχέση με κάποιο σημείο (κ', $\lambda^*(P_{ik'})$), κ'< κ.

Για το τμήμα των κορυφών αριστερά της *Min*(i,j) παρατηρείται η $\lambda^*(P_{kj})$ μειώνεται, οπότε αρκούν οι κορυφές για τις οποίες αυξάνεται η τιμή της συνάρτησης $\lambda^*(P_{ik})$. Τα σημεία (k, $\lambda^*(P_{ik})$) που βρίσκονται βόρειο-δυτικά (ΒΔ) αναφορικά με κάποιο σημείο (κ', $\lambda^*(P_{ik'})$) με κ'>κ απορρίπτονται. Με τον τρόπο αυτό προκύπτει η (ΒΔ) δομή_σκάλας.

Τα σημεία τομής των δομών_σκάλας με την $\lambda^*(P_{kj})$ είναι τα σημεία, όπου παρατηρείται η μικρότερη τιμή της Min_Max τριγωνοποίησης και αποτελούν τα Min_Max ζεύγη. Η απάντηση βρίσκεται μετά από έλεγχο στα σημεία δεξιά και αριστερά της τομής, στο δεξιό και στο αριστερό τμήμα.

Απεικονίζεται σχηματικά στο σχήμα που ακολουθεί.

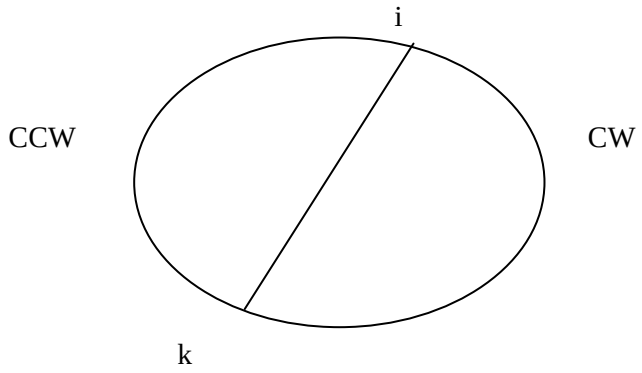



Σχήμα 3-17 Τα δύο Σημεία Τομής της Δομής Σκάλας με την Καμπύλη $\lambda^*(P_{kj})$

Ο χρόνος εύρεσης της ελάχιστης τιμής *Min*(i,j) πραγματοποιείται σε O(logn) χρόνο με δυαδική αναζήτηση στον πίνακα $\lambda^*(P_{kj})$ για κάθε διαγώνιο *ij*. Για την εύρεση των σημείων τομής των δομών_σκάλας με την συνάρτηση $\lambda^*(P_{kj})$ χρειάζονται δύο το πολύ δυαδικές αναζητήσεις στο αριστερό και στο δεξιό τμήμα του διαστήματος [*Left*(i,j),*Right*(i,j)].

Όσον αφορά τη μελέτη για τη χωρική πολυπλοκότητα του αλγορίθμου σημειώνεται πως για κάθε κορυφή, *i*, απαιτείται:

- ένας πίνακας για τη καταχώρηση των τιμών των $\lambda^*(P_{ik})$
- ένας πίνακας για τη καταχώρηση των τιμών των $\lambda^*(P_{kj})$
- δύο πίνακες που καταχωρούν τις φιλτραρισμένες εκδοχές των δύο εκδοχών των δομών_σκάλας, της **ΒΑ** εκδοχής και της **ΒΔ** δομής και για τους δύο παραπάνω πίνακες και
- ένας πίνακας για την καταχώρηση των τιμών των *Min*(i,j).




Σχήμα 3-18 Για κάθε Κορυφή i και για κάθε k Καταχωρεί τις Τιμές των $\lambda^*(P_{ik})$ σε Φορά Ωρολογιακή και Ανθωρολογιακή

Για τη μελέτη της χρονικής πολυπλοκότητας του *Min_Max* αλγορίθμου σημειώνονται τα παρακάτω:

Ο χρόνος εύρεσης και ταξινόμησης των κορυφών με την ένδειξη *Min*(i,j) για το σύνολο των $n^2$ διαγωνίων είναι της τάξης $O(n^2 \log n)$.

Η εύρεση των λύσεων των υπό-προβλημάτων των $P_{ik}$ και $P_{kj}$ είναι της τάξης $O(n^2)$.

Για το τμήμα στα δεξιά της *Min*(i,j) κορυφής και αριστερά της *Right*(i,j) κορυφής οι διαγώνιοι με αρχή την κορυφή *i*, εξετάζονται με φθίνουσα σειρά, αναφορικά με τη θέση της κορυφής *Min*(i,j) για την κάθε διαγώνιο *ij*.

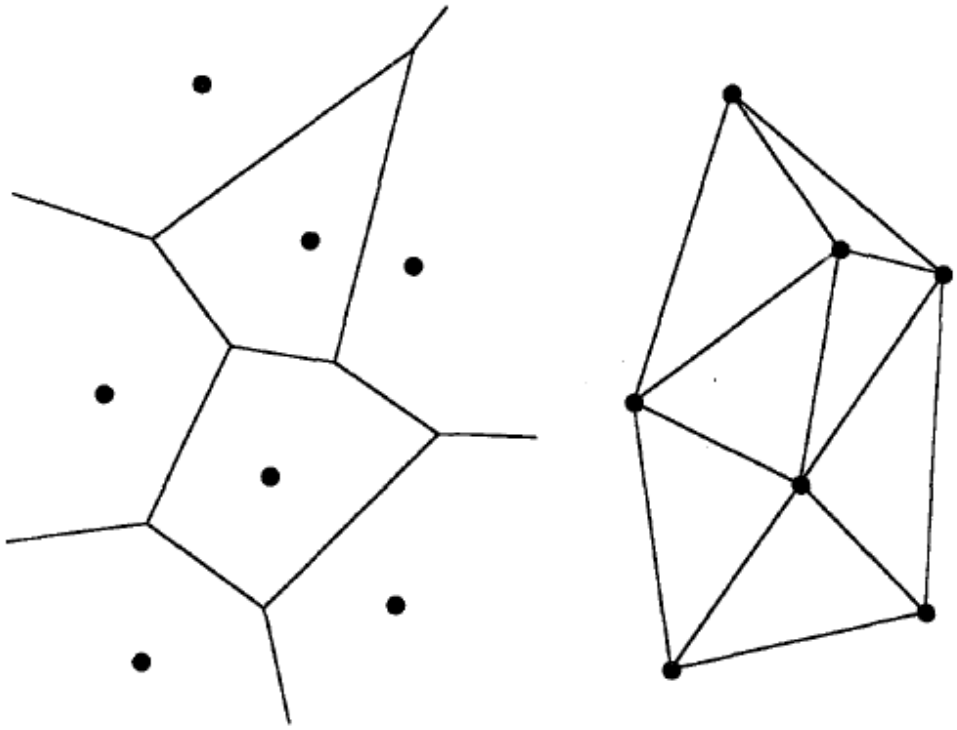
Για την πρώτη διαγώνιο εξετάζονται όλα τα σημεία από το (*i*-1) έως το *Min*(i,j) απορρίπτοντας τα σημεία **BA**. Για την επόμενη διαγώνιο, η θέση της δικής της *Min*(i,j) βρίσκεται στα αριστερά της προηγούμενης, οπότε εξετάζονται τα σημεία που βρίσκονται ανάμεσα στις δύο θέσεις της *Min*(i,j) και ενημερώνεται το αριστερό άκρο της δομής σκάλας.
Αν κάποιο σημείο από την αρχή της δομής σκάλας «καλύπτεται» από ένα νέο σημείο στα αριστερά βγαίνει από τη δομή. Για το λόγο αυτό χρειάζεται να διατρέχεται η δομή της σκάλας από την αρχή και να ελέγχονται τα σημεία της. Κάθε σημείο στη σκάλα μπαίνει και βγαίνει από αυτή ακριβώς μια φορά. Έτσι, η ενημέρωση της δομής της σκάλας για όλες τις διαγώνιους μιας κορυφής απαιτεί χρόνο γραμμικό και για το σύνολο των κορυφών απαιτείται χρόνος και χώρος $O(n^2)$.

Η εύρεση του σημείου τομής με την $\lambda^*(P_{kj})$ γίνεται με δυαδική αναζήτηση σε $O(\log n)$ χρόνο. Η απόρριψη των σημείων από τη δομή της σκάλας γίνεται σε λογαριθμικό χρόνο $O(\log n)$.

*3.3.2. Η Delauny διαίρεση σε τρίγωνα*

Ένας σημαντικός αλγόριθμος χωρισμού σε τρίγωνα είναι κι αυτός της *Delaunay* διαίρεσης σε τρίγωνα, ο οποίος μεγιστοποιεί την ελάχιστη γωνία τριγώνου στο σύνολο των δυνατών τριγωνοποιήσεων ενός συνόλου σημείων. Όταν καμία τετράδα σημείων δεν ορίζει ομοκυκλικά σημεία, τότε η *Delaunay* διαίρεση είναι μοναδική.

Η *Delaunay* διαίρεση σε τρίγωνα αποτελεί το δυϊκό γράφημα *G*, του διαγράμματος *Voronoi*, *V*(P), του συνόλου των σημείων. Προκύπτει αντιστοιχίζοντας σε κάθε περιοχή του διαγράμματος *Voronoi* μια κορυφή (μπορεί να συμπίπτει με κάποιο σημείο του αρχικού συνόλου). Δύο κορυφές του *G* συνδέονται με ακμή, εάν οι αντίστοιχες περιοχές *Voronoi* έχουν κοινή ακμή.



Σχήμα 3.19 Το διάγραμμα Voronoi ενός συνόλου σημείων και η αντίστοιχη *Delaunay* διαίρεση σε τρίγωνα

Το γράφημα *G* παράγει μια διαίρεση του κυρτού περιβλήματος του συνόλου των σημείων σε τρίγωνα. Οι ακραίες ακμές της διαμέρισης ορίζουν το κυρτό περίβλημα του συνόλου των σημείων, εφόσον περιέχονται και σημεία στο εσωτερικό του κυρτού περιβλήματος.

Κάποιες από τις σημαντικότερες εφαρμογές του διαγράμματος *Voronoi* και κατ' επέκταση και της *Delaunay* διαίρεσης σε τρίγωνα αφορά στην εύρεση:

- των κοντινότερων γειτόνων,
- του μέγιστου κενού κύκλου,
- του ελάχιστου συνδετικού δένδρου (*minimum spanning tree)* ενός συνόλου σημείων, το οποίο είναι ένα δέντρο ελαχίστου μήκους, που συνδέει όλα τα σημεία του συνόλου, έστω *S*.

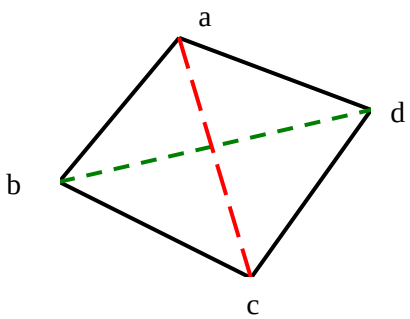
Αποδεικνύεται ότι το ελάχιστο συνδετικό δένδρο *MST(S)* είναι υποσύνολο της *Delaunay* διαίρεσης σε τρίγωνα.

Ο γενικός αλγόριθμος της *Delaunay* διαίρεσης σε τρίγωνα τρέχει σε $O(n^2)$, ενώ στην περίπτωση ενός απλού πολυγώνου μπορεί να υπολογιστεί σε $O(n \log n)$ χρόνο, που είναι και ο βέλτιστος χρόνος υπολογισμού. Υπάρχουν διάφοροι αλγόριθμοι υπολογισμού της *Delaunay* διαίρεσης σε τρίγωνα που εφαρμόζουν τεχνικές όπως η «διαίρει-και-βασίλευε», η τεχνική της σάρωσης γραμμής, η τεχνική «edge-flipping», αυξητικοί αλγόριθμοι ή τέλος με γεωμετρικούς μετασχηματισμούς.

Παρακάτω παρουσιάζονται ικανές και αναγκαίες συνθήκες για να είναι μια τριγωνοποίηση *Delaunay* τριγωνοποίηση καθώς και κάποιες ιδιότητες αυτής.

Ιδιότητες *Delaunay* διαίρεσης σε τρίγωνα

Μια τριγωνοποίηση είναι *Delaunay* εάν και μόνο εάν είναι *ισογώνια* (*equiangular*) τοπικά. [9]. Μια τριγωνοποίηση είναι τοπικά *ισογώνια*, εάν για οποιαδήποτε δύο τρίγωνα της τριγωνοποίησης, τα οποία έχουν μια κοινή ακμή και των οποίων η ένωση είναι ένα αυστηρά κυρτό τετράπλευρο, τότε η αντικατάσταση της κοινής τους ακμής με την άλλη διαγώνιο του τετραπλεύρου δεν αυξάνει την ελάχιστη εκ των έξι γωνιών στα δύο τρίγωνα, που συνθέτουν το τετράπλευρο. Και απεικονίζεται σχηματικά παρακάτω.



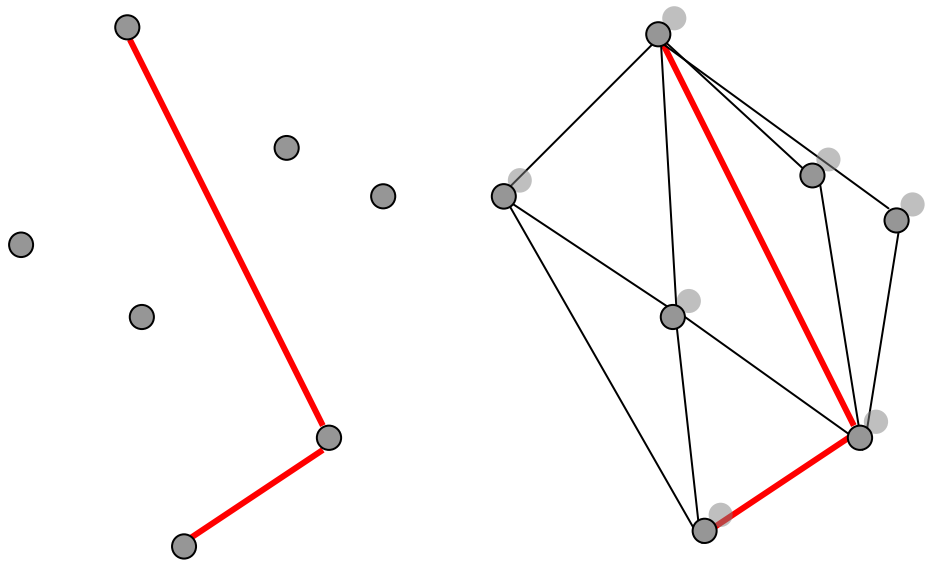

Σχήμα 3.20 Η εναλλαγή των Δύο Διαγωνίων σε μια Delaunay Διαίρεση

Αποδεικνύεται, επίσης, ότι η *Delaunay* διαίρεση σε τρίγωνα ελαχιστοποιεί την ακτίνα του μέγιστου περιγεγραμμένου κύκλου του συνόλου των τριγώνων στο σύνολο των δυνατών τριγωνοποιήσεων [1].

Μια παραλλαγή της γενικής μορφής της *Delaunay* διαίρεσης σε τρίγωνα αποτελεί και η *Constrained Delaunay* (CDT) διαίρεση σε τρίγωνα [7]. Γενικά, η *Constrained* τριγωνοποίηση ενός επίπεδου γραφήματος, G, προϋποθέτει η τριγωνοποίηση να περιέχει τις ακμές του G και υπολογίζεται σε O(nlogn) χρόνο. Η *Constrained Delaunay* τριγωνοποίηση είναι μια *Constrained* τριγωνοποίηση, που είναι όσο «κοντά» γίνεται στην *Delaunay* διαίρεση σε τρίγωνα. Αυτό μεταφράζεται στο εξής: κάθε ακμή του επίπεδου γραφήματος G ανήκει στην τριγωνοποίηση T και για κάθε εναπομένουσα ακμή, *e*, της T, υπάρχει κύκλος *c* με τις ακόλουθες ιδιότητες:
Τα άκρα της *e* ανήκουν στη περιφέρεια του κύκλου c.

Εάν κάποια κορυφή v του G ανήκει στο εσωτερικό ενός κύκλου *c*, τότε ένα από τα άκρα της *e* δεν μπορεί να τη «δει», με την έννοια ότι αν ενωθεί με την *v* με ένα ευθύγραμμο τμήμα, αυτό θα τέμνεται από κάποια άλλη ακμή της τριγωνοποίησης T.

Αυτό απεικονίζεται σχηματικά παρακάτω.



Σχήμα 3.21 Ένα γράφημα G και η αντίστοιχη Constrained Delaunay Τριγωνοποίηση

Ισχύει, από τον ορισμό, ότι εάν το G δεν έχει ακμές τότε η *Constrained Delaunay* τριγωνοποίηση ταυτίζεται με την (*unconstrained*) *Delaunay* τριγωνοποίηση.

### *3.3.3. Τριγωνοποίηση Ελαχίστου κόστους*

Στη δεδομένη περίπτωση βελτιστοποίησης σε κάθε ακμή τριγώνου ανατίθεται μία τιμή κόστους. Έτσι, η ζητούμενη τριγωνοποίηση αναμένεται να έχει την ελάχιστη τιμή κόστους στο σύνολο των ακμών κάθε τριγώνου.

Αν η συνάρτηση κόστους θεωρείται ότι έχει την ίδια τιμή, έστω ίση με τη μονάδα (1), για κάθε ακμή, τότε καταλήγει κάθε χωρισμός σε τρίγωνα να θεωρείται βέλτιστος, εφόσον όλοι οι πιθανοί χωρισμοί σε τρίγωνα περιέχουν τον ίδιο αριθμό ακμών, που είναι ίσος με 3$n$-3 ακμές, και άρα δεν τίθεται θέμα βέλτιστου αλγορίθμου.

Η άλλη περίπτωση είναι το κόστος κάθε ακμής να είναι είτε 1 είτε 2. Τότε το πρόβλημα της εύρεσης της βέλτιστης τριγωνοποίησης με το ελάχιστο κόστος θεωρείται NP-πλήρες. Κάτι τέτοιο στηρίζεται στην απόδειξη του ότι η απόφαση για το εάν υπάρχει μια τριγωνοποίηση δεδομένου ότι κάποιες ακμές δεν επιτρέπονται είναι πρόβλημα NP-πλήρες [2]. Κι αυτό γιατί είναι το ίδιο με το να αναθέτεις τιμή κόστους ίση με 1 σε ακμές που επιτρέπονται και τιμή κόστους ίση με 2 σε ακμές που δεν επιτρέπονται και να αναζητείς μια οποιαδήποτε διαίρεση σε τρίγωνα κόστους 3$n$-3.

Στη τελευταία περίπτωση, όπου η τιμή κόστους κάθε ακμής είναι ίση με το μήκος της ακμής, εμπίπτουν οι αλγόριθμοι *Minimum Length* ή *Minimum Weight Triangulations.*

# ΚΕΦΑΛΑΙΟ 4. ΥΛΟΠΟΙΗΣΗ – ΠΕΙΡΑΜΑΤΙΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

---



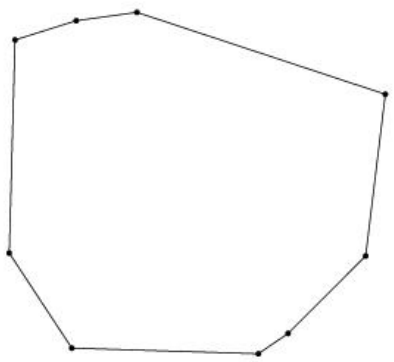
---

## 4.1. Είσοδος – Έξοδος

Η είσοδος του αλγορίθμου είναι ένα αρχείο δεδομένων το οποίο περιέχει στη πρώτη σειρά το πλήθος, *n*, των κορυφών του πολυγώνου και ακολουθούν στις επόμενες σειρές οι συντεταγμένες των κορυφών του πολυγώνου, ως προς τους άξονες x και y, έχοντας μια σειρά για κάθε κορυφή.

Ένα παράδειγμα αρχείου εισόδου είναι το παρακάτω αρχείο poly1b.txt:
Input polygon (file: poly1b.txt)

Number of vertices: 9
Coordinates of vertices
0: -47.09707 39.47092
1: -31.34144 44.44158
2: -15.66927 46.60365
3: 48.36231 25.41847
4: 43.29646 -16.38718
5: 23.30955 -36.40876
6: 15.64576 -41.67173
7: -32.47101 -40.21086



8: -48.58783 -15.64145

Η έξοδος του αλγορίθμου περιέχει τα τρίγωνα που προκύπτουν με το βέλτιστο χωρισμό, ως προς το κριτήριο, που έχει θέσει ο χρήστης στην αρχή της εκτέλεσης του προγράμματος σε δύο μορφές. Στην πρώτη μορφή το κάθε τρίγωνο αναπαρίσταται από τους δείκτες των τριών κορυφών του (στο αρχικό πολύγωνο, όπως διαβάζοναι από το αρχείο). Η δεύτερη μορφή εμφανίζει ένα αρχείο σε μορφή *postscript*, όπου εμφανίζει το αρχικό πολύγωνο και τις διαγώνιους που προσθέτει η τριγωνοποίηση.

Ενδεικτικά παρουσιάζεται παρακάτω μια ενδεικτική έξοδος στη περίπτωση που εφαρμόζεται το κριτήριου του εμβαδού για την Max_Min περίπτωση για το παραπάνω πολύγωνο, που είναι καταχωρημένο στο αρχείο poly1b.txt (με έντονη-πλάγια γραφή, οι απαντήσεις του χρήστη).
Give the name of the file containing the polygon: ***poly1b.txt***

Input polygon (file: poly1b.txt)

Number of vertices: 9
Coordinates of vertices
0: -47.09707 39.47092
1: -31.34144 44.44158
2: -15.66927 46.60365
3: 48.36231 25.41847
4: 43.29646 -16.38718
5: 23.30955 -36.40876
6: 15.64576 -41.67173
7: -32.47101 -40.21086
8: -48.58783 -15.64145

The vertices of P are in Clockwise order.

Input polygon (file: poly1b.txt)

Number of vertices: 9
Coordinates of vertices
0: -47.09707 39.47092
1: -31.34144 44.44158

2: -15.66927 46.60365
3: 48.36231 25.41847
4: 43.29646 -16.38718
5: 23.30955 -36.40876
6: 15.64576 -41.67173
7: -32.47101 -40.21086
8: -48.58783 -15.64145

Choose one of the following criteria:
1. area
2. inradius
3. circumradius
4. ratio of circumradius over inradius
5. angle
Type the number to the left of the desired criterion to select it
or any other number to exit: ***1***

Choose the desired optimization algorithm
by typing the corresponding number:
1. MaxMin
2. MinMax
choice: ***1***

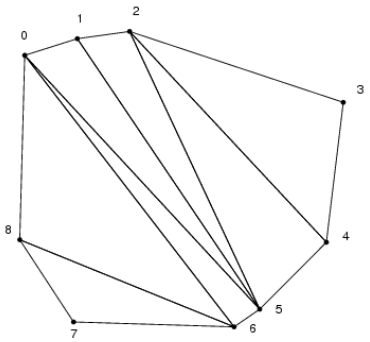
Processing

The values stored in the array SubPr are:

--- --- 21.918 235.229 557.345 692.630 476.037 476.037 476.037
476.037 --- --- 235.229 557.345 692.630 453.623 453.623 453.623
476.037 476.037 --- --- 1392.101 1219.790 420.667 420.667 420.667
476.037 476.037 476.037 --- --- 367.070 170.989 170.989 268.775
476.037 476.037 476.037 476.037 --- --- 24.125 132.216 268.775
476.037 476.037 476.037 476.037 476.037 --- --- 132.216 268.775
579.328 579.328 579.328 579.328 579.328 476.037 --- --- 579.328
462.432 462.432 462.432 462.432 557.345 692.630 476.037 --- ---
--- 430.460 430.460 430.460 557.345 692.630 476.037 476.037 ---

The indices stored in the array SubPr are:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -- | -- | 1 | 1 | 1 | 1 | 5 | 6 | 6 |
| 4 | -- | -- | 2 | 2 | 2 | 5 | 6 | 6 |
| 3 | 4 | -- | -- | 3 | 4 | 5 | 6 | 6 |
| 4 | 4 | 4 | -- | -- | 4 | 5 | 6 | 4 |
| 5 | 5 | 5 | 2 | -- | -- | 5 | 5 | 5 |
| 6 | 0 | 1 | 2 | 0 | -- | -- | 6 | 6 |
| 8 | 0 | 1 | 2 | 2 | 0 | -- | -- | 7 |
| 8 | 0 | 1 | 2 | 8 | 8 | 8 | -- | -- |
| -- | 0 | 1 | 2 | 0 | 0 | 0 | 6 | -- |




Triangle ( 0, 6, 8)
Triangle ( 0, 5, 6)
Triangle ( 0, 1, 5)
Triangle ( 1, 2, 5)
Triangle ( 2, 4, 5)
Triangle ( 2, 3, 4)
Triangle ( 6, 7, 8)

Επίσης δημιουργείται ένα αρχείο με τις διαφορετικές τιμές των λύσεων βελτιστοποίησης στα υπό-προβλήματα $P_{ik}$, $P_{kj}$ και της τιμής του κριτηρίου για το τρίγωνο **Δ**ikj για κάθε τιμή του k, k = *i*+1 .. *i*-1.

Τα υπό-προβλήματα $P_{ij}$ ορίζονται για $|i - j| \geq 2$, ενώ διαφορετικά δεν έχουν τιμή, αφού πρόκειται για την περίπτωση όπου το πολύγωνο εκφυλίζεται σε μια ακμή.

### 4.2. Δομές

Η δομή ***Vertex*** χρησιμοποιείται για την καταχώριση των κορυφών του πολυγώνου:

```
typedef struct vertex
{
   double x;
   double y;
```

```
} Vertex;
```

Στη δομή αποθηκεύονται οι συντεταγμένες x και y τύπου *double* της κάθε κορυφής.

Ο πίνακας ***Pol*** τα στοιχεία του οποίου είναι τύπου **Vertex** αποθηκεύει το σύνολο των κορυφών του πολυγώνου και είναι ένας πίνακας δυναμικής καταχώρησης *n* θέσεων μνήμης.

Η δομή ***Sub_sol*** καταχωρεί τη λύση κάθε υπό-προβλήματος και αποτελείται από τις μεταβλητές *index* και *value*:

```
typedef struct subsol
{
  int index;
  double value;
}Sub_sol;
```

Η μεταβλητή *index* καταχωρεί τη θέση της κορυφής, στον πίνακα *Pol*, με την οποία συνδέεται η διαγώνιος <*i, j*>. Η μεταβλητή *value* αναπαριστά την τιμή του κριτηρίου για τη λύση του δεδομένου υπό-προβλήματος. Για την αποθήκευση των λύσεων κάθε δυνατού συνδυασμού δύο κορυφών δηλώνεται ο δισδιάστατος πίνακας, *SubPr*, $n^2$ θέσεων, όπου η τιμή των *indices* για τα σημεία της κυρίας διαγωνίου είναι ίση με -1. Οι τιμές του *SubPr* αλλάζουνε κατά τη διάρκεια εκτέλεσης του αλγορίθμου και συνιστούν τις λύσεις του δυναμικού προγραμματισμού του αλγορίθμου.

### 4.3. Επιλεγμένες Συναρτήσεις

#### *4.3.1. Η συνάρτηση read_P*

Η συνάρτηση *read_P* διαβάζει το αρχείο, το όνομα του οποίου δίνει ο χρήστης κατά την εκτέλεση του προγράμματος και καταχωρεί τις συντεταγμένες των κορυφών του στον πίνακα *Pol*, που αναφέρθηκε παραπάνω. Στην περίπτωση, που δεν δίνεται σωστό όνομα ή δεν υπάρχει ελεύθερος χώρος στη μνήμη εμφανίζει αντίστοιχο μήνυμα.

```
void read_P(void)
{
    FILE *fptr;
```

```
    int i=0;
    printf("Give the name of the file containing the polygon: ");
    scanf("%s",fname);
    printf("%s\n",fname);
    if ((fptr = fopen(fname,"r")) == NULL)
   {
       printf("Cannot open file for reading, Aborting\n");
       exit(1);
   }
   fscanf(fptr,"%d",&n);
/* n : number of polygon vertices */
/* Allocate space for the array Pol[n] which will store */
/* the vertices of the polygon                          */
Pol = (Vertex *) calloc(n,sizeof(Vertex));
if (Pol==NULL)
{
  printf("Out of memory\n");
  exit(1);
}
/* read the coordinates of the vertices */
for (i=0; i<n; i++)
  fscanf(fptr,"%lf %lf",&Pol[i].x,&Pol[i].y);
fclose(fptr);
}
```

*4.3.2. Η συνάρτηση f*

Η συνάρτηση *f* υπολογίζει την τιμή του κριτηρίου για ένα τρίγωνο τριών κορυφών: Α, Β και C. Σύμφωνα με την επιλογή του χρήστη το κριτήριο μπορεί να είναι ένα από τα: εμβαδόν τριγώνου, μήκος ακτίνας περιγεγραμμένου και εγγεγραμμένου κύκλου τριγώνου, λόγος μηκών ακτίνας περιγεγραμμένου προς ακτίνας εγγεγραμμένου κύκλου (ρ) και μέγιστης ή ελάχιστης γωνίας. Και τα τρία χαρακτηριστικά στηρίζονται σε τύπους υπολογισμού, που ήδη έχουν αναφερθεί προηγουμένως. Εκείνο, που αξίζει να σημειωθεί είναι για το κριτήριο της γωνίας η συνάρτηση επιστρέφει την τιμή του αντίθετου συνημίτονου της γωνίας.

```
double f(Vertex A, Vertex B, Vertex C, int criterion, int opt_type)
{
   double r, R, E, radii, a, b, c, x, y, h, z;
   Vertex m, p, q, s;

   E = abs_val(signed_area(A,B,C));
   if (criterion == 1)
     return(E);
   else
  {
      a = sqrt((C.x-B.x)*(C.x-B.x) + (C.y-B.y)*(C.y-B.y));
      b = sqrt((A.x-C.x)*(A.x-C.x) + (A.y-C.y)*(A.y-C.y));
      c = sqrt((A.x-B.x)*(A.x-B.x) + (A.y-B.y)*(A.y-B.y));
     if (criterion == 2)
     {    /* inradius */
           r = (2.0 * E) / (a + b + c);
           return(r);
     }
      else if (criterion == 3)
      {       /* circumradius */
          R=      (a * b * c) / (4.0 * E);
          return(R);
      }
     else if (criterion == 4)
     {    /* ratio of circumradius over inradius */
          radii = (a * b * c * (a+b+c)) / (8.0 * E * E);
          return(radii);
     }
     else if (criterion == 5)
     {    /* angle */
          if (opt_type == MaxMin)
          {
             x = find_min(a,b,c);
          }
        else
        {
```

```
        x = find_max(a,b,c);
    }
    if (x > a - EPSILON && x < a + EPSILON)
    {                                       /* x ~= a */
    p=A;
    q=B;
    s=C;
    y=b;
    h=(2.0 * E) / c;
    }
    else if (x > b - EPSILON && x < b + EPSILON)
    {                                       /* x ~= b */
    p=B;
    q=C;
    s=A;
    y=c;
    h=(2.0 * E) / a;
    }
    else
    {                                       /* x ~= c */
    p=C;
    q=A;
    s=B;
    y=a;
    h=(2.0 * E) / b;
    }
    z = sqrt(y * y - h * h);
    m.x = p.x + (q.y - p.y);
    m.y = p.y - (q.x - p.x);
    /* return the value of -cos(angle) */
    if (turn(m,p,q) * turn(m,p,s) > 0)
    return(-z/y);
    else
    return(z/y);
    }
    }
```

}

*4.3.3. Η συνάρτηση Check_P*

Η συνάρτηση *Check_P* ελέγχει την κυρτότητα του πολυγώνου και στη περίπτωση, που δεν είναι κυρτό τερματίζει την εκτέλεση του κώδικα. Επίσης, ελέγχει την περίπτωση που το αρχείο δεδομένων περιέχει λιγότερα από ή ακριβώς τρία σημεία, όπου εμφανίζει το αντίστοιχο μήνυμα στην οθόνη και τερματίζει την εκτέλεση.

```
int check_P(int n, Vertex *v)
{
        int i, t;
        if (n <= 2)
        {
          printf("Degenerate polygon (fewer than 3 vertices).");
          printf("  Aborting...\n");
           exit(-1);
        }
        else if (n == 3)
        {
           printf("The given polygon is a triangle.  Aborting...\n");
           exit(-1);
        }
        else
        {
           for(i=1; i<n-1; i++)
          {
              if ((t = turn(v[i-1], v[i], v[i+1])) != STRAIGHT)
              break;
          }
           if (i == n-1)
          {
                printf("Degenerate polygon (vertices collinear).");
                printf("  Aborting...\n");
                exit(-1);
```

```
            }
          }
         else
         {
           for(; i<n-1; i++)
           {
             if (t * turn(v[i-1], v[i], v[i+1]) < 0)
               return(0);
         }
         if (t * turn(v[n-2], v[n-1], v[0]) < 0    || t * turn(v[n-1], v[0], v[1]) < 0)
            return(0);
        else
            return(1);
}
}
}
```

*4.3.4. Η συνάρτηση Check_CW_ordering*

Η συνάρτηση *Check_CW_ordering* ελέγχει τη φορά των κορυφών του πολυγώνου έτσι ώστε εκείνη να είναι *ωρολογιακή* (*clockwise*). Σε διαφορετική περίπτωση καλείται η συνάρτηση *repair_CCW_ordering*, η οποία αναδιατάσσει τη θέση των κορυφών του πίνακα *Pol*, έτσι ώστε η δεύτερη να πάει τελευταία, η τελευταία να γίνει δεύτερη κ.ο.κ , έως ότου όλες οι κορυφές να διαταχθούν σε ωρολογιακή φορά. Κάτι τέτοιο γίνεται γιατί ο αλγόριθμος προϋποθέτει ότι οι κορυφές συναντώνται κατά την ωρολογιακή φορά.

```
void check_CW_ordering(void)
{
        int i, min=0;
        /* find the index (stored in min) of leftmost lowermost vetex of P */
        for(i=1; i<n; i++)
        {
           if (Pol[min].x > Pol[i].x)
              min = i;
           else if (Pol[min].x > Pol[i].x - EPSILON)
```

```
            if (Pol[i].y > Pol[min].y)
                min = i;
          }
        /* check turn at min to determine vertex ordering */
      if(turn(Pol[my_mod(min-1,n)], Pol[min], Pol[my_mod(min+1,n)]) !=     RIGHT)
       {
            printf("\nThe vertices of P are in CounterClockwise order.\n");
            repair_CCW_ordering();
            printf("The ordering of the vertices of P has been changed");
            printf(" to Clockwise.\n");
        }
       else
           printf("\nThe vertices of P are in Clockwise order.\n");
}

void repair_CCW_ordering(void)
{
          int    i, j;
          Vertex temp;
          for (i=0, j=n-1; i < j; i++, j--)
          {
            temp = Pol[i];
            Pol[i] = Pol[j];
            Pol[j] = temp;
          }
}
```

*4.3.5. Η συνάρτηση signed_area*

Η συνάρτηση *signed_area* η οποία υπολογίζει για τρεις κορυφές, που περνιόνται ως ορίσματα το προσημασμένο εμβαδόν του τριγώνου που σχηματίζεται από τις τρεις κορυφές. Ο τύπος υπολογισμού στον οποίο βασίζεται είναι ο τύπος της ορίζουσας, που δίνεται από τη σχέση (9), που υπολογίζει το εμβαδόν ενός τριγώνου βασισμένη στις συντεταγμένες των τριών κορυφών.

```
double signed_area(Vertex i, Vertex j, Vertex k)
```

```
{
        double a;
        a = 0.5 * ((i.x * j.y) - (i.y * j.x) + (i.y * k.x) - (i.x * k.y)
           + (j.x * k.y) - (j.y * k.x));
        return(a);
}
```

*4.3.6. Η συνάρτηση turn*

Η συνάρτηση *turn* υπολογίζει για τις τρεις κορυφές που δέχεται ως ορίσματα, την μεταξύ τους στροφή. Έτσι, αν οι A,B και C σχηματίζουν δεξιά γωνία η τιμή της *signed_area* για τις ίδιες κορυφές είναι αρνητική, διαφορετικά είναι θετική και στην περίπτωση που επιστρέφει μηδέν οι κορυφές A, B και C είναι συνευθειακές.

```
int turn(Vertex i, Vertex j, Vertex k)
{
        double h;
        h = signed_area(i,j,k);
        if (h < - EPSILON)
             return (RIGHT);
        else if (h > EPSILON)
             return (LEFT);
        else
            return(STRAIGHT);
}
```

*4.3.7. Η συνάρτηση my_mod*

Η συνάρτηση *my_mod* φροντίζει ο δείκτης της κορυφής που περνιέται ως όρισμα να είναι σε μορφή *modulo* n, οπότε να έχει τιμή που κυμαίνεται από 0 έως n-1.

```
int my_mod(int v, int k)
{
```

```
        int h;
        if ( v < 0)
            h = k + v;
        else if ( v > k-1)
           h = v - k;
        else
           h = v;
        return(h);
}
```

*4.3.8. Η συνάρτηση two_zone*

Η συνάρτηση *two_zone* με δείκτες δύο δείκτες i, j ελέγχει εάν το υπό-πολύγωνο ενδιάμεσα των κορυφών i και j είναι 2-*zone* με βάση ότι έχει αναλυθεί πιο πάνω, οπότε εάν είναι 2-*zone* επιστρέφει true, διαφορετικά επιστρέφει false.

```
int two_zone(int i,int j)
{
        Vertex p;

        p.x = Pol[i].x + (Pol[my_mod(j-1,n)].x - Pol[j].x);
        p.y = Pol[i].y + (Pol[my_mod(j-1,n)].y - Pol[j].y);
        if (turn(p,Pol[i],Pol[j]) * turn(p,Pol[i],Pol[my_mod(i+1,n)]) >= 0)
           return(TRUE);
        else
           return(FALSE);
}
```

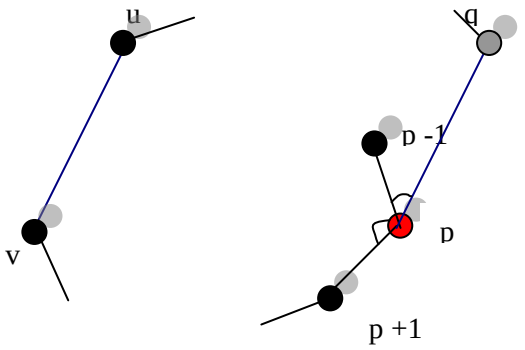
*4.3.9. Η συνάρτηση is_top*

Η συνάρτηση *is_top* με ορίσματα τις κορυφές p, v και u ελέγχει εάν η κορυφή p είναι η κορυφή Top(v,u). Για τον έλεγχο αυτό θεωρείται στοιχείο, q, το οποίο προκύπτει από μια μετατόπιση του σημείου p στην κατεύθυνση ευθείας παράλληλης προς την

ακμή <*u*, v>, και το οποίο δίνεται από τη σχέση $q = p + (v - u)$.

```
int is_top(int p,int v,int u)
{
        int prev,next;
        Vertex q;
        q.x = Pol[p].x + Pol[v].x - Pol[u].x;
        q.y = Pol[p].y + Pol[v].y - Pol[u].y;
        prev= my_mod(p-1,n);
        next= my_mod(p+1,n);
        if (turn(q, Pol[p], Pol[next]) == 0
         || turn(q, Pol[p], Pol[prev]) * turn(q, Pol[p], Pol[next]) > 0)
           return(TRUE);
        else
          return(FALSE);
}
```

Σχηματικά, απεικονίζεται παρακάτω. Έτσι, αν η στροφή που σχηματίζεται από την *q*, *p* και *p*-1 είναι ίδια με εκείνη που σχηματίζεται από την *q*, *p* και *p*+1 ή τα *p*, *q* και *p*+1 είναι συνευθειακά, τότε η *p* είναι η πιο απομακρυσμένη κορυφή για την ακμή <*u*, *v*>, διαφορετικά δεν είναι. Στο σχήμα απεικονίζεται η περίπτωση που η *p* είναι η Top κορυφή.



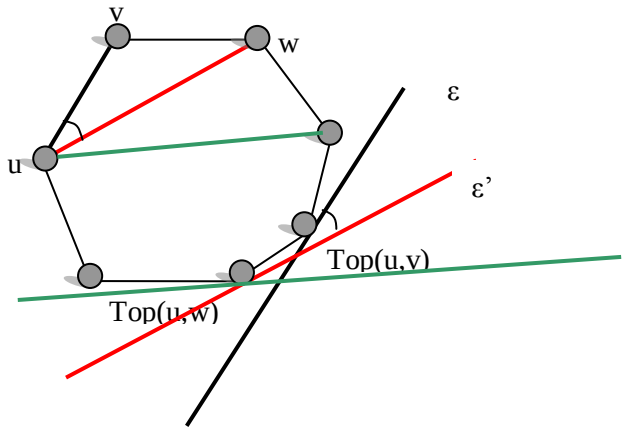

Σχήμα 4.1 Υπολογισμός της κορυφής Top για την ακμή <v, u>.

*4.3.10. Η συνάρτηση Calculate_Top*

Η συνάρτηση αυτή καλείται από την *Calculate_Top* η οποία υπολογίζει τις Top κορυφές κάθε διαγωνίου i j για i=0..n-1 και j=i+1..i-2. Εξετάζει πρώτα τις διαγώνιους που είναι γειτονικές σε μια κορυφή, έστω u, διαδοχικά τη μία μετά την άλλη και συνεχίζει για διαγώνιους γειτονικές στην επόμενη κορυφή της u. Η συνάρτηση υλοποιείται σε $O(n^2)$ χρόνο.

```
void Calculate_Top(void)
{
        int i,j,p,h;
        for (i=0; i<n; i++)
        {
            j = my_mod(i+1,n);
            p = my_mod(j+1,n);
            for (h=0; h < n-2; h++)
           {
              while(!is_top(p,j,i))
              {
                  p= my_mod(p+1,n);
               }
               Top[j][i]=p;
               j= my_mod(j+1,n);
           }
        }
}
```

Η λογική του να εξετάζει τις διαγώνιους που έχουν κοινή κορυφή, την πρώτη κορυφή, με ωρολογιακή φορά, στηρίζεται στο ότι η Top(u,w) κορυφή της επόμενης της uv διαγωνίου, θα είναι είτε η Top(u, v) είτε κάποια επόμενη κορυφή. Με τον τρόπο αυτό γλιτώνει τον διεξοδικό έλεγχο από τη αρχή όλων των n-1 κορυφών για την εύρεση της Top διαγωνίων με κοινή την πρώτη κορυφή, οπότε ο χρόνος από $O(n^3)$ που θα ήτανε μειώνεται στο $O(n^2)$. Απεικονίζεται ένα αντίστοιχο σχήμα παρακάτω, όπου φαίνεται ο υπολογισμός της Top(u,v) και της Top(u,w), για ένα παράδειγμα πολυγώνου 7 κορυφών. Παρατηρεί κάποιος ότι η γωνία μεταξύ των διαγωνίων (u,v) και (u,w) είναι ίδια με εκείνη μεταξύ των ευθειών ε // uv, η οποία διέρχεται από την Top(u,v)) και ε' //uw, η οποία διέρχεται από την Top(u,w). Και ομοίως ισχύει για όλες τις επόμενες διαγώνιους, που έχουν, ως πρώτη κορυφή, κάθε κορυφή του πολυγώνου.




Σχήμα 4.2 Προσδιορισμός κορυφών Top δύο γειτονικών διαγώνιων μιας κορυφής, u, του πολυγώνου.

*4.3.11. Η συνάρτηση Calculate_MaxCW*

Η επόμενη συνάρτηση είναι η συνάρτηση *Calculate_MaxCW*, η οποία υπολογίζει τις συναρτήσεις MaxCW(i) και MaxCCW(i) για κάθε κορυφή του πολυγώνου, δηλαδή για i=0..n-1. Ο υπολογισμός αυτών στηρίζεται στους παρακάτω τύπους, που είναι οι εξής:
MaxCW(i) = Top(i, i+1) ή Top(i, i+1) +1 και
MaxCCW(i) = Top(i, i-1).

*4.3.12. Η συνάρτηση process*

Η συνάρτηση *process* υλοποιεί το κομμάτι που αφορά το δυναμικό προγραμματισμό του αλγορίθμου, υπολογίζει τις λύσεις των υποπροβλημάτων (i, j) και τις οποίες λύσεις καταχωρεί σε ένα δισδιάστατο πίνακα, ονόματι SubPr, όπου η λύση στο υπό-πολύγωνο των κορυφών από την i έως την j, δίνεται από την τιμές των πεδίων value και index του στοιχείου SubPr[i,j], όπως και την τελική λύση.

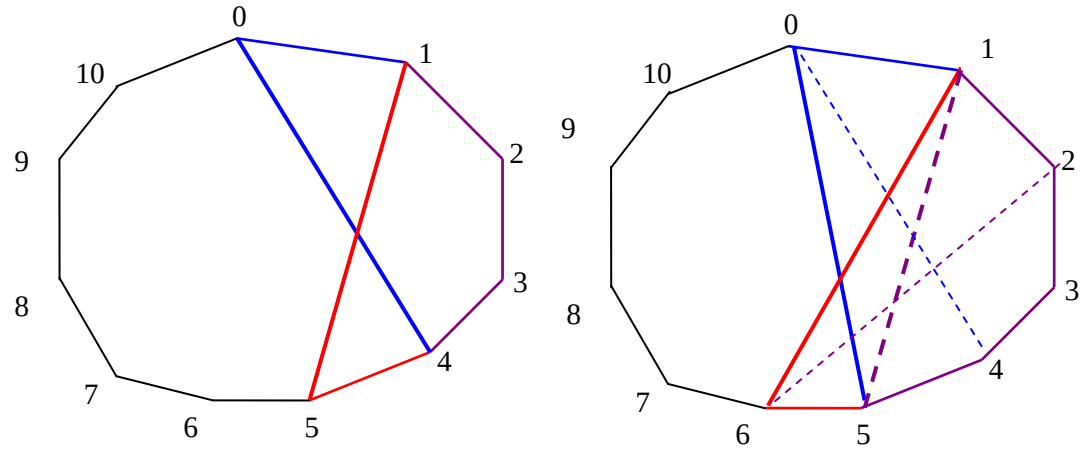
```
void process(int n, Vertex *Pol, int criterion, int opt_type)
{
        int i, j, k, step, cur_ind;
```

```
        double cur_val, x, y, z;
        for (step=2; step < n; step++)
         for (i=0; i < n; i++)
        {
           j = my_mod(i+step, n);
           cur_ind = -1;
           for (k=my_mod(i+1,n); k != j; k=my_mod(k+1,n))
          {
              printf("\nTriangle: v[%d]-v[%d]-v[%d]\n", i, k, j);
              y = f(Pol[i],Pol[k],Pol[j],criterion,opt_type);
              if (k != my_mod(i+1,n))
                x = SubPr[i][k].value;
              else
                 x = y;
              if (k != my_mod(j-1,n))
                   z = SubPr[k][j].value;
             else
                 z = y;
             if (opt_type == MaxMin)
             {
               x = find_min(x,y,z);
               if (cur_ind < 0 || cur_val < x)
               {
                 cur_ind = k;
                 cur_val = x;
               }
         else
        {                                              /* Min Max */
            x = find_max(x,y,z);
            if (cur_ind < 0 || cur_val > x)
           {
                cur_ind = k;
                cur_val = x;
            }
        }
}
```

```
SubPr[i][j].value = cur_val;
SubPr[i][j].index = cur_ind;
}
}
```

Στη συνάρτηση περιέχονται τρεις βρόγχοι: ο εξωτερικός βρόγχος αφορά το βήμα και κυμαίνεται από *step* = 2 .. n-1 και καθορίζει το μέγεθος του υπό-πολυγώνου, το οποίο εξετάζεται και το οποίο χτίζεται σταδιακά. Για αρχική τιμή *step* = 2 δίνεται η λύση σε όλα τα συνοριακά τρίγωνα, που σχηματίζονται στο σύνορο του P από διαδοχικές κορυφές.

Η αρχική κορυφή κάθε υπό-πολυγώνου καθορίζεται από το δείκτη του επόμενου βρόγχου, *i*, ο οποίος κυμαίνεται από *i* = 0 ..n-1. Παρακάτω, παρουσιάζονται στιγμιότυπα του προβλήματος για τιμές των *step*= 3, 4 και *i*= 0, 1, για ένα παράδειγμα πολυγώνου 11 κορυφών.




Σχήμα 4.3 Στιγμιότυπα λύσεων υπό-πολυγώνων για step=3,4 και i=0,1

Η τελική κορυφή σε κάθε υπό-πολύγωνο $P_{ij}$ , *j* ισούται με *j* = *i* + *step* οπότε παίρνει τιμή στο εσωτερικό των δύο βρόγχων. Στη μεταβλητή cur_ind και cur_val, αντίστοιχα καταχωρείται η τιμή του δείκτη της κορυφής και της τιμής του κριτηρίου για τη βέλτιστη λύση στο υπό-πρόβλημα ij. Εάν ο αλγόριθμος βελτιστοποίησης είναι ο MaxMin θεωρεί κάθε φορά την ελάχιστη τιμή από τις τιμές των προβλημάτων ik, kj και της τιμής του τριγώνου ikj και θεωρεί ως λύση τη μέγιστη τιμή στο τέλος. Εάν ο αλγόριθμος είναι ο MinMax υπολογίζει τη μέγιστη τιμή εκ των ik, kj και του τριγώνου ikj και θεωρεί ως λύση την ελάχιστη τιμή αυτής στο τέλος των επαναλήψεων. Ο βρόγχος για την αναζήτηση της σωστής κορυφής k είναι της τάξης O(n) και συνολικά η συνάρτηση έχει πολυπλοκότητα χρόνου $O(n^3)$.

*4.3.13. Η συνάρτηση get_triangle*

Η συνάρτηση get_triangle δέχεται ως ορίσματα τους δείκτες δύο κορυφών, i, j, που αναπαριστούν τους δείκτες του υπό-προβλήματος το οποίο λύνουν για να τυπώσει το τρίγωνο **Δ**ikj και καλείται αναδρομικά για το τμήμα ik και το τμήμα kj. Τυπώνει έως ότου τα i, j που δέχεται ως ορίσματα είναι διαδοχικές κορυφές.

```
void get_triangle(int i, int j)
{
        int k;

        k = SubPr[i][j].index;
        printf("Triangle (%2d, %2d, %2d)\n", i, k, j);
        if (k != my_mod(i+1,n))
          get_triangle(i, k);
        if (k != my_mod(j-1,n))
           get_triangle(k, j);
}
```

*4.3.14. Η συνάρτηση collect_triangle*

Η συνάρτηση collect_triangle, η οποία καλεί την get_triangle με ορίσματα τις κορυφές 0 και n-1, οπότε τυπώνει τα τρίγωνα της λύσης ολόκληρου του πολυγώνου.

```
void colect_triangles()
{
        get_triangle(0,n-1);
}
```

*4.3.15. Η συνάρτηση study_values*

Η συνάρτηση study_values δημιουργεί ένα αρχείο με τα αποτελέσματα των τιμών του αριστερού υπό-προβλήματος $P_{ik}$, του δεξιού υπο-προβλήματος $P_{kj}$ και της τιμής του κριτηρίου για το τρίγωνο **Δ**ikj για κάθε δυνατή τιμή του δείκτη της κορυφής κ και για κάθε ακμή του πολυγώνου.

Τα αποτελέσματα του αρχείου είναι χρήσιμα για την μελέτη των αποτελεσμάτων και την εξαγωγή κάποιων συμπερασμάτων για τη συμπεριφορά των τιμών των $P_{ik}$ και $P_{kj}$. Για την οπτικοποίηση των αποτελεσμάτων μπορούν να γίνουν και διαγράμματα με την απόδοση των εκάστοτε τιμών.

```
void study_values(int n, Vertex *Pol, int criterion, int opt_type)
{
        int  i, j, k, prev_j;
        double y;
        char vname[21];
        FILE *fptr;

        sprintf(vname, "values_%d_%d.%s", criterion, opt_type, fname);
        if ((fptr = fopen(vname,"w")) == NULL)
        {
           printf("Cannot open file for writing, Aborting\n");
           exit(1);
        }
        for (i=0; i < n; i++)
        {
          j = my_mod(i+n-1, n);
          prev_j = my_mod(j-1,n);
          k = my_mod(i+1,n);
          fprintf(fptr, "\nEdge %d-%d\n", i, j);
          y = f(Pol[i],Pol[k],Pol[j],criterion,opt_type);
          fprintf(fptr, " %2d              %8.3lf   %8.3lf\n", k, y, SubPr[k][j].value);
          for ( ; (k=my_mod(k+1,n)) != prev_j; )
         {
          y = f(Pol[i],Pol[k],Pol[j],criterion,opt_type);
          fprintf(fptr, " %2d   %8.3lf   %8.3lf   %8.3lf\n",
          k, SubPr[i][k].value, y, SubPr[k][j].value);
        }
        y = f(Pol[i],Pol[prev_j],Pol[j],criterion,opt_type);
        fprintf(fptr, " %2d   %8.3lf   %8.3lf\n", prev_j, SubPr[i][prev_j].value, y);
        }
        fclose(fptr);
```

}

### 4.4. Ενδεικτικές Εκτελέσεις

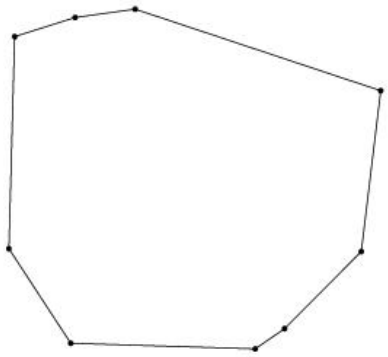
Ακολουθούν ενδεικτικές εκτελέσεις για τα πολυγώνα poly1 και poly2, για κάθε κριτήριο και κάθε έναν από τους δύο αλγορίθμους Max_Min και Min_Max.

#### *4.4.1. Πολύγωνο Poly1*

Number of vertices: 9

Coordinates of vertices

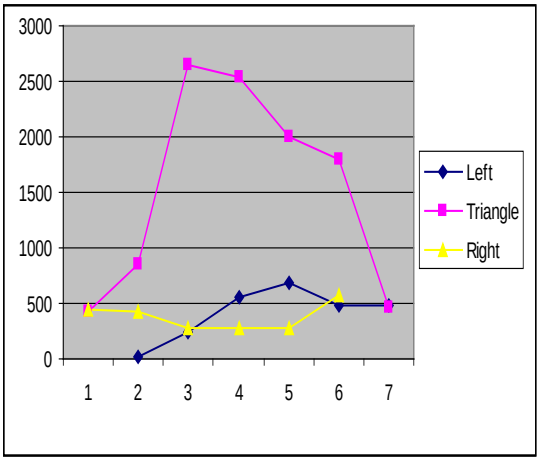
0: -47.09707 39.47092
1: -31.34144 44.44158
2: -15.66927 46.60365
3: 48.36231 25.41847
4: 43.29646 -16.38718
5: 23.30955 -36.40876
6: 15.64576 -41.67173
7: -32.47101 -40.21086
8: -48.58783 -15.64145



-----Max_Min Εμβαδών

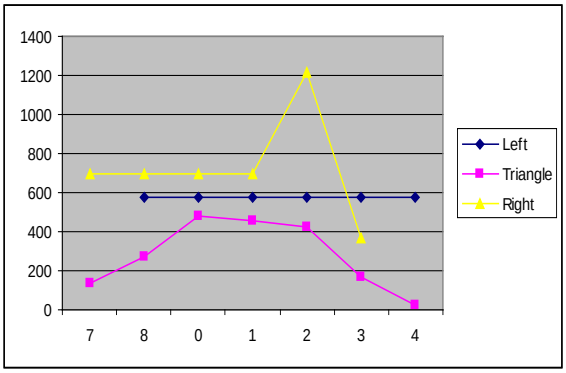
Edge 0-8

| | Left | Triangle | Right |
|---|---|---|---|
| 1 | | 430,46 | 453,623 |
| 2 | 21,918 | 860,714 | 420,667 |
| 3 | 235,229 | 2640,971 | 268,775 |
| 4 | 557,345 | 2532,536 | 268,775 |
| 5 | 692,63 | 1996,697 | 268,775 |
| 6 | 476,037 | 1789,435 | 579,328 |
| 7 | 476,037 | 462,432 | |



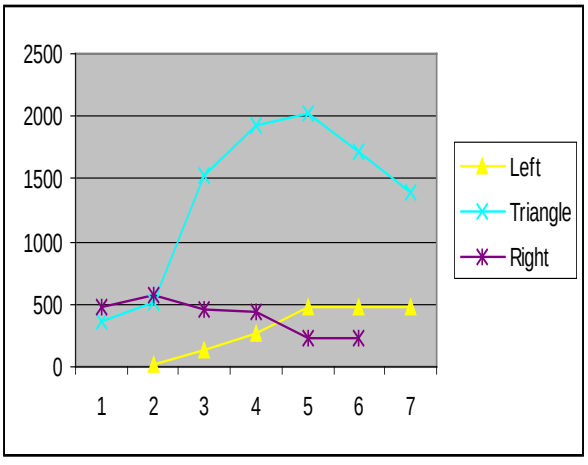

Edge 4 - 3

| | Left | Triangle | Right |
|---|---|---|---|
| 5 | | 367,07 | 476,037 |
| 6 | 24,125 | 513,934 | 579,328 |
| 7 | 132,216 | 1523,41 | 462,432 |
| 8 | 268,775 | 1922,53 | 430,46 |
| 0 | 476,037 | 2030,965 | 235,229 |
| 1 | 476,037 | 1714,218 | 235,229 |
| 2 | 476,037 | 1392,101 | |




Edge 6 - 5

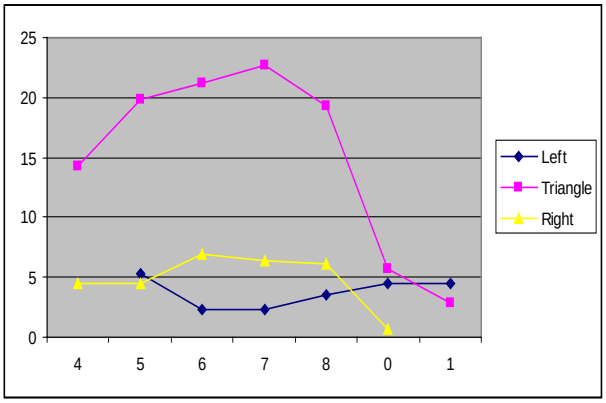
| | Left | Triangle | Right |
|---|---|---|---|
| 7 | | 132,216 | 692,63 |
| 8 | 579,328 | 268,775 | 692,63 |
| 0 | 579,328 | 476,037 | 692,63 |
| 1 | 579,328 | 453,623 | 692,63 |
| 2 | 579,328 | 420,667 | 1219,79 |
| 3 | 579,328 | 170,989 | 367,07 |
| 4 | 579,328 | 24,125 | |



----Max_Min Inradius

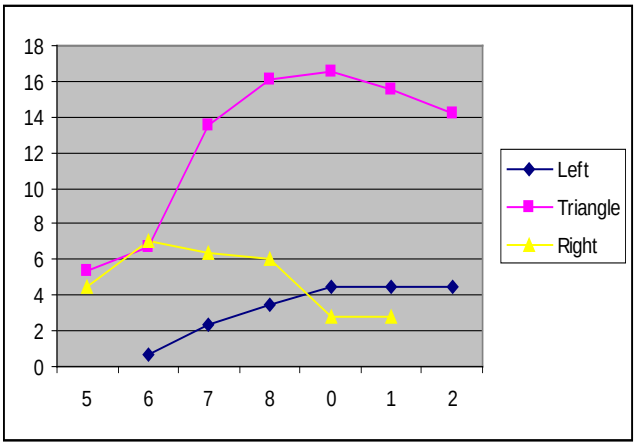
Edge 3-2

| | Left | Triangle | Right |
|---|---|---|---|
| 4 | | 14,217 | 4,426 |
| 5 | 5,354 | 19,878 | 4,426 |
| 6 | 2,27 | 21,162 | 6,991 |
| 7 | 2,333 | 22,75 | 6,417 |
| 8 | 3,503 | 19,26 | 6,08 |
| 0 | 4,42 | 5,722 | 0,679 |
| 1 | 4,426 | 2,848 | |



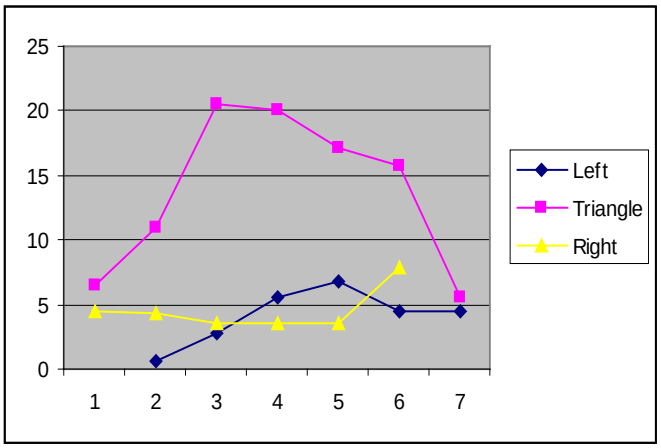

Edge 4-3

| | Left | Triangle | Right |
|---|---|---|---|
| 5 | | 5,354 | 4,426 |
| 6 | 0,643 | 6,665 | 6,991 |
| 7 | 2,333 | 13,502 | 6,417 |
| 8 | 3,503 | 16,069 | 6,08 |
| 0 | 4,42 | 16,589 | 2,848 |
| 1 | 4,426 | 15,56 | 2,848 |
| 2 | 4,426 | 14,217 | |




Edge 0-8

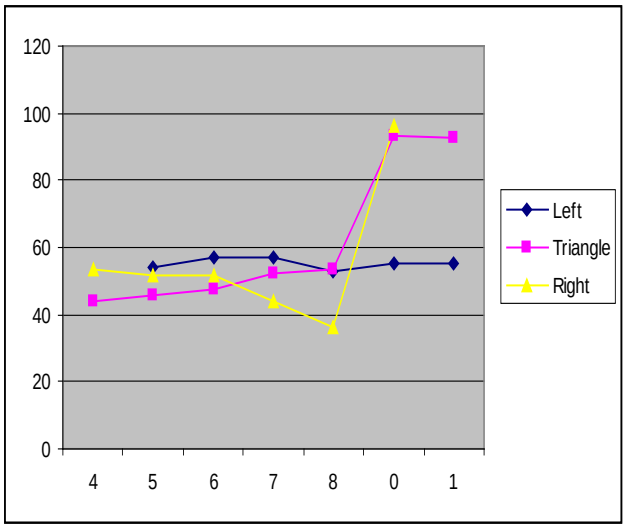
| | Left | Triangle | Right |
|---|---|---|---|
| 1 | | 6,417 | 4,426 |
| 2 | 0,679 | 10,911 | 4,322 |
| 3 | 2,848 | 20,56 | 3,503 |
| 4 | 5,619 | 19,998 | 3,503 |
| 5 | 6,753 | 17,104 | 3,503 |
| 6 | 4,426 | 15,765 | 7,891 |
| 7 | 4,426 | 5,587 | |



---Max_Min Circumradius

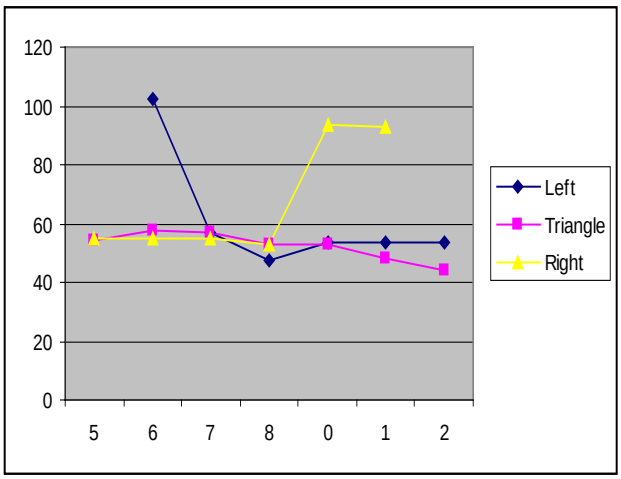
Edge 3-2

| | Left | Triangle | Right |
|---|---|---|---|
| 4 | | 44,01 | 53,535 |
| 5 | 54,128 | 45,953 | 51,839 |
| 6 | 57,29 | 47,258 | 51,643 |
| 7 | 57,102 | 52,492 | 44,251 |
| 8 | 52,915 | 53,385 | 36,338 |
| 0 | 54,987 | 93,416 | 96,077 |
| 1 | 54,987 | 92,925 | |



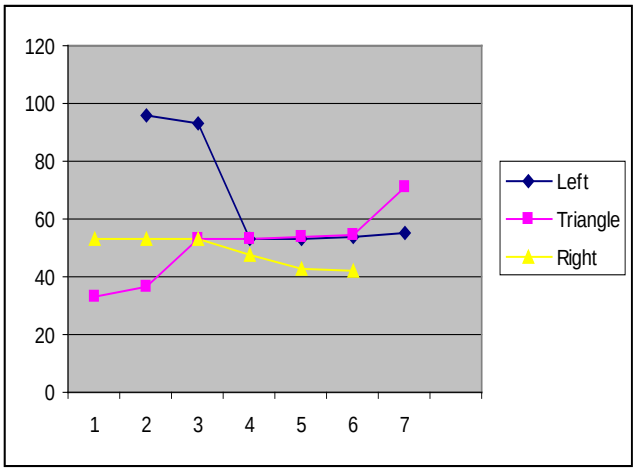

Edge 4-3

| | Left | Triangle | Right |
|---|---|---|---|
| 5 | | 54,128 | 54,987 |
| 6 | 102,119 | 57,29 | 54,987 |
| 7 | 56,983 | 57,15 | 54,987 |
| 8 | 47,75 | 52,978 | 53,019 |
| 0 | 53,535 | 53,147 | 93,416 |
| 1 | 53,535 | 48,456 | 92,925 |
| 2 | 53,535 | 44,01 | |




Edge 0-8

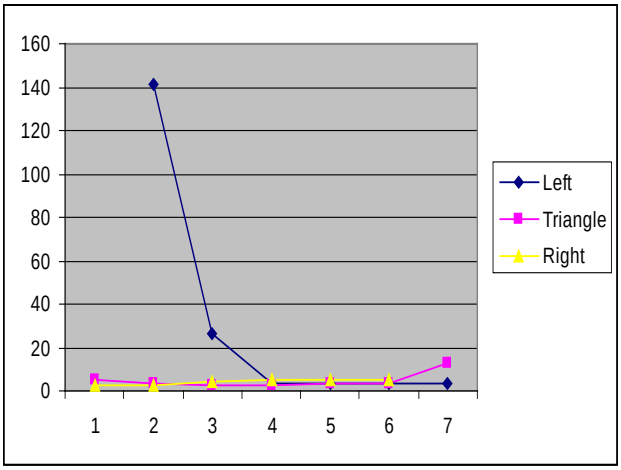
| | Left | Triangle | Right |
|---|---|---|---|
| 1 | | 33,067 | 53,358 |
| 2 | 96,077 | 36,338 | 52,915 |
| 3 | 93,416 | 53,019 | 52,915 |
| 4 | 53,147 | 53,139 | 47,75 |
| 5 | 53,268 | 53,474 | 42,935 |
| 6 | 53,811 | 54,757 | 42,306 |
| 7 | 54,987 | 70,952 | |



--- Max_Min Radii Ratio

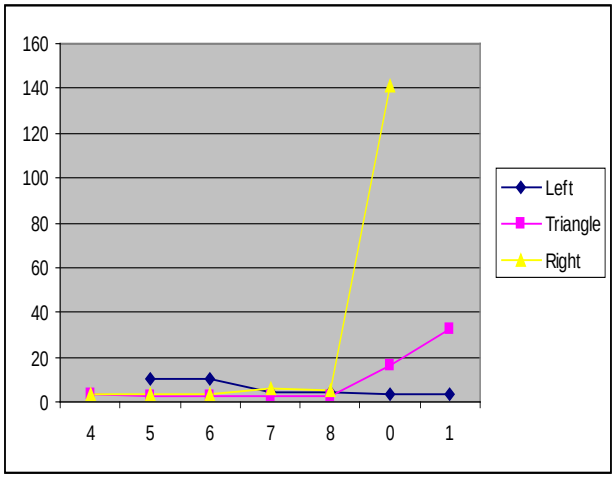
Edge 0-8

| | Left | Triangle | Right |
|---|---|---|---|
| 1 | | 5,153 | 2,78 |
| 2 | 141,516 | 3,331 | 2,772 |
| 3 | 26,292 | 2,579 | 4,151 |
| 4 | 3,204 | 2,657 | 5,361 |
| 5 | 3,204 | 3,126 | 5,361 |
| 6 | 3,204 | 3,473 | 5,361 |
| 7 | 3,188 | 12,699 | |



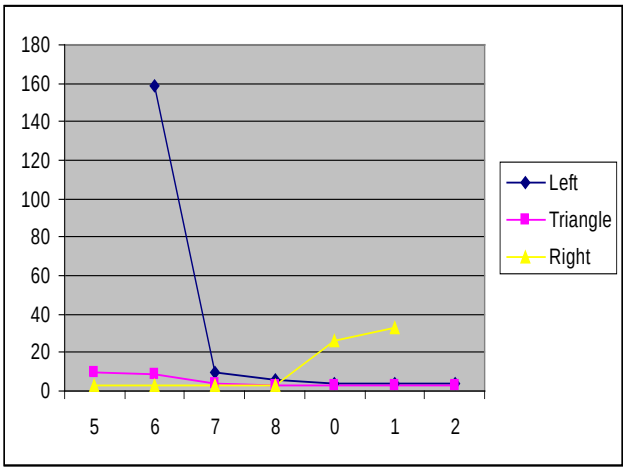

Edge 3-2

| | Left | Triangle | Right |
|---|---|---|---|
| 4 | | 3,096 | 3,473 |
| 5 | 10,109 | 2,312 | 3,473 |
| 6 | 10,109 | 2,233 | 3,473 |
| 7 | 4,233 | 2,307 | 5,852 |
| 8 | 4,151 | 2,772 | 5,153 |
| 0 | 3,204 | 16,324 | 141,516 |
| 1 | 3,204 | 32,632 | |




Edge 4-3

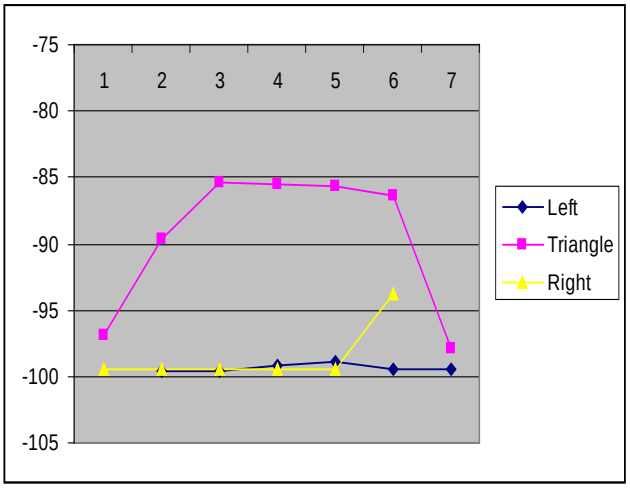
| | Left | Triangle | Right |
|---|---|---|---|
| 5 | | 10,109 | 2,772 |
| 6 | 158,85 | 8,596 | 2,772 |
| 7 | 9,487 | 4,233 | 2,772 |
| 8 | 5,361 | 3,297 | 2,772 |
| 0 | 3,473 | 3,204 | 26,292 |
| 1 | 3,473 | 3,114 | 32,632 |
| 2 | 3,473 | 3,096 | |



--- Max_Min Γωνία

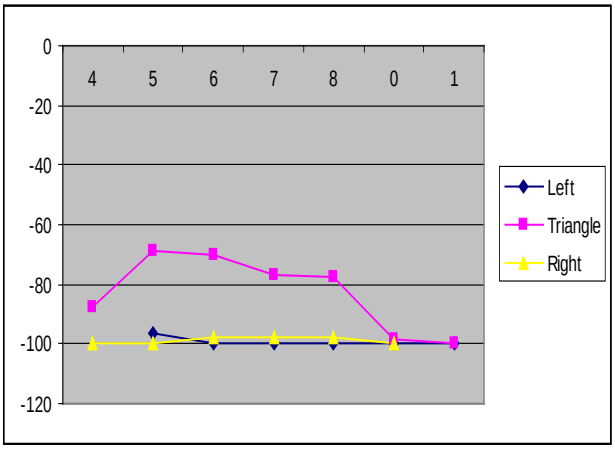
Edge 0-8

| | Left | Triangle | Right |
|---|---|---|---|
| 1 | | -96,829 | -99,461 |
| 2 | -99,66 | -89,631 | -99,461 |
| 3 | -99,637 | -85,421 | -99,461 |
| 4 | -99,152 | -85,492 | -99,461 |
| 5 | -98,823 | -85,688 | -99,461 |
| 6 | -99,519 | -86,403 | -93,776 |
| 7 | -99,516 | -97,833 | |



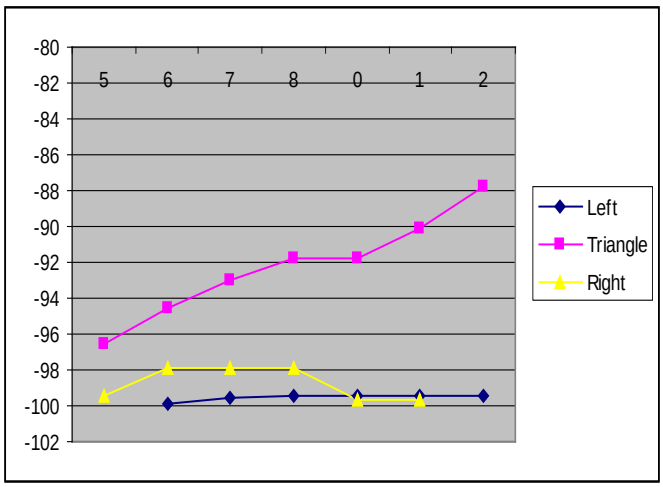

Edge 3-2

| | Left | Triangle | Right |
|---|---|---|---|
| 4 | | -87,812 | -99,461 |
| 5 | -96,525 | -68,784 | -99,461 |
| 6 | -99,764 | -70,056 | -97,867 |
| 7 | -99,516 | -76,635 | -97,867 |
| 8 | -99,461 | -77,523 | -97,867 |
| 0 | -99,461 | -98,501 | -99,66 |
| 1 | -99,461 | -99,637 | |




Edge4-3

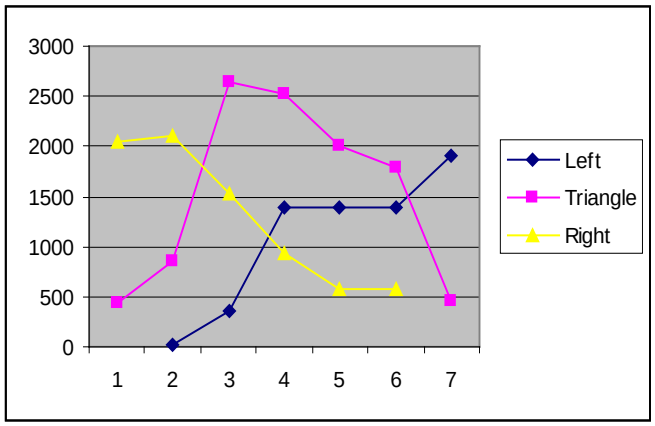
| | Left | Triangle | Right |
|---|---|---|---|
| 5 | | -96,525 | -99,461 |
| 6 | -99,896 | -94,502 | -97,867 |
| 7 | -99,516 | -92,966 | -97,867 |
| 8 | -99,461 | -91,763 | -97,867 |
| 0 | -99,461 | -91,817 | -99,637 |
| 1 | -99,461 | -90,065 | -99,637 |
| 2 | -99,461 | -87,812 | |



---Min_Max εμβαδών

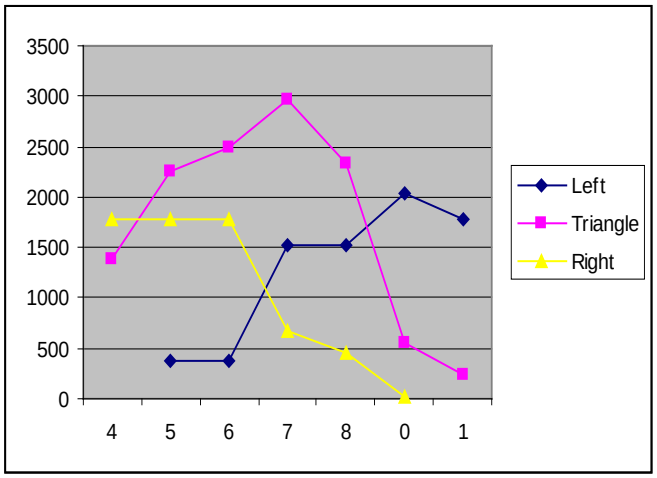
Edge 0-8

| | Left | Triangle | Right |
|---|---|---|---|
| 1 | | 430,46 | 2037,426 |
| 2 | 21,918 | 860,714 | 2100,889 |
| 3 | 347,951 | 2640,971 | 1523,41 |
| 4 | 1392,101 | 2532,536 | 927,287 |
| 5 | 1392,101 | 1996,697 | 579,328 |
| 6 | 1392,101 | 1789,435 | 579,328 |
| 7 | 1906,331 | 462,432 | |



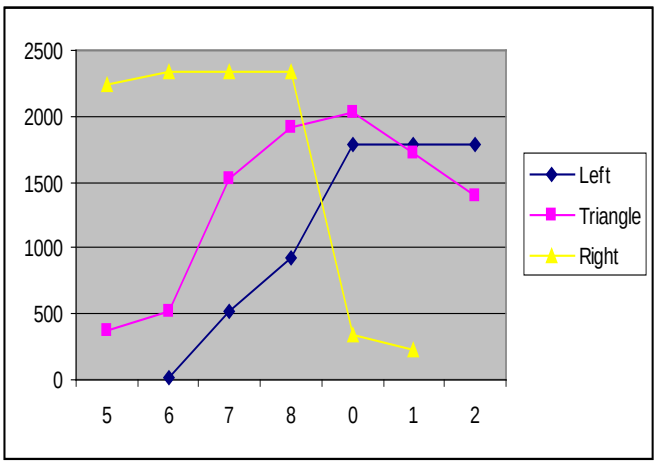

Edge 3-2

| | Left | Triangle | Right |
|---|---|---|---|
| 4 | | 1392,101 | 1789,435 |
| 5 | 367,07 | 2244,821 | 1789,435 |
| 6 | 367,07 | 2494,499 | 1789,435 |
| 7 | 1523,41 | 2957,409 | 664,069 |
| 8 | 1523,41 | 2341,519 | 452,172 |
| 0 | 2030,965 | 561,262 | 21,918 |
| 1 | 1789,435 | 235,229 | |




Edge 4-3

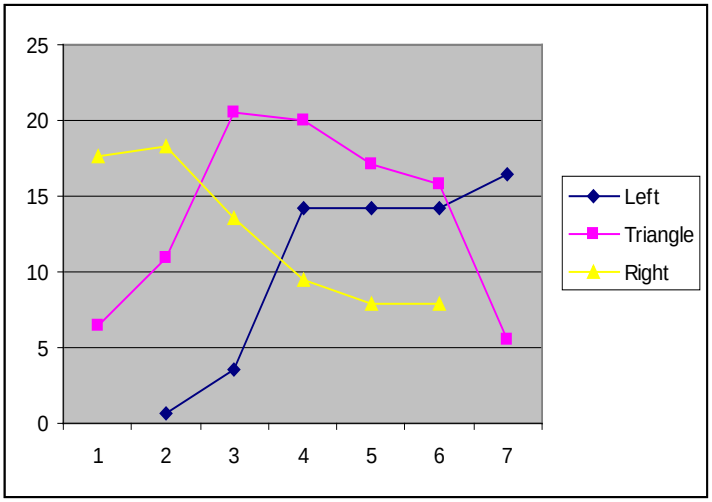
| | Left | Triangle | Right |
|---|---|---|---|
| 5 | | 367,07 | 2244,821 |
| 6 | 24,125 | 513,934 | 2341,519 |
| 7 | 520,411 | 1523,41 | 2341,519 |
| 8 | 927,287 | 1922,53 | 2341,519 |
| 0 | 1789,435 | 2030,965 | 347,951 |
| 1 | 1789,435 | 1714,218 | 235,229 |
| 2 | 1789,435 | 1392,101 | |



---Min_Max inradius

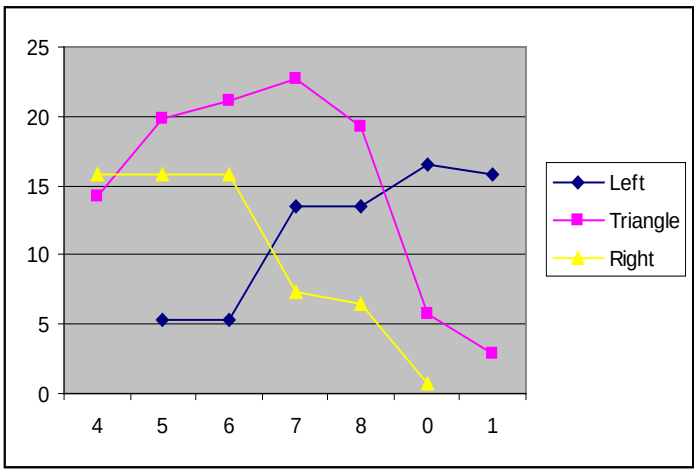
Edge 0-8

| | Left | Triangle | Right |
|---|---|---|---|
| 1 | | 6,417 | 17,648 |
| 2 | 0,679 | 10,911 | 18,25 |
| 3 | 3,57 | 20,56 | 13,502 |
| 4 | 14,217 | 19,998 | 9,51 |
| 5 | 14,217 | 17,104 | 7,891 |
| 6 | 14,217 | 15,765 | 7,891 |
| 7 | 16,454 | 5,587 | |



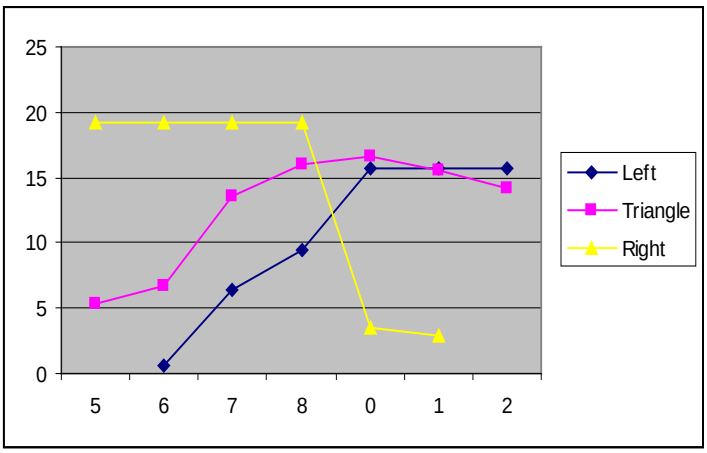

Edge 3-2

| | Left | Triangle | Right |
|---|---|---|---|
| 4 | | 14,217 | 15,765 |
| 5 | 5,354 | 19,878 | 15,765 |
| 6 | 5,354 | 21,162 | 15,765 |
| 7 | 13,502 | 22,75 | 7,29 |
| 8 | 13,502 | 19,26 | 6,417 |
| 0 | 16,589 | 5,722 | 0,679 |
| 1 | 15,765 | 2,848 | |




Edge 4-3

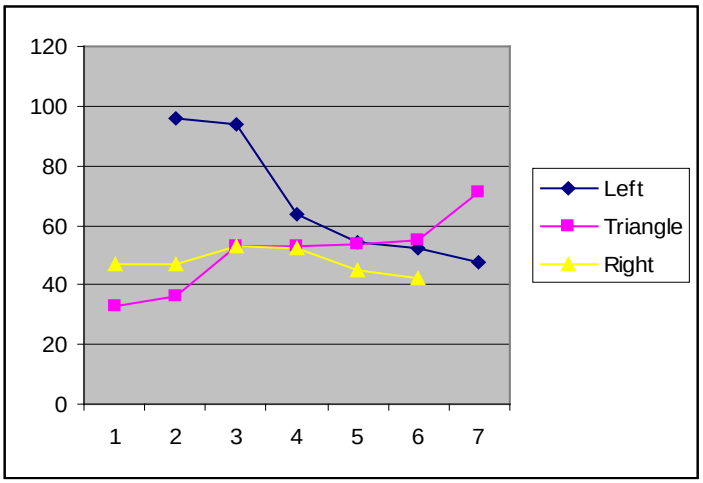
| | Left | Triangle | Right |
|---|---|---|---|
| 5 | | 5,354 | 19,26 |
| 6 | 0,643 | 6,665 | 19,26 |
| 7 | 6,361 | 13,502 | 19,26 |
| 8 | 9,51 | 16,069 | 19,26 |
| 0 | 15,765 | 16,589 | 3,57 |
| 1 | 15,765 | 15,56 | 2,848 |
| 2 | 15,765 | 14,217 | |



---Min_Max Circumradius

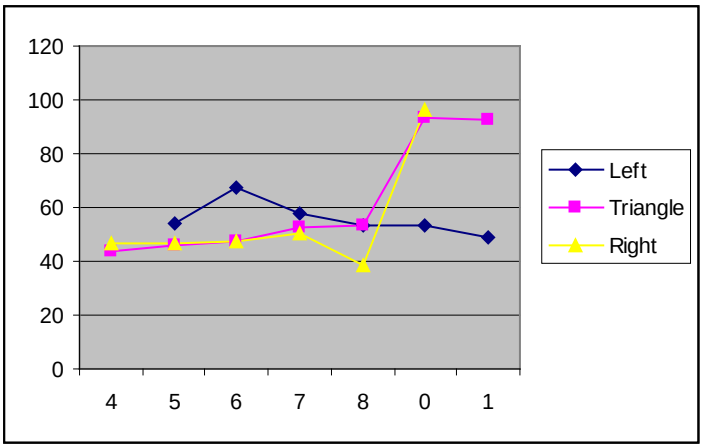
Edge 0-8

| | Left | Triangle | Right |
|---|---|---|---|
| 1 | | 33,067 | 46,837 |
| 2 | 96,077 | 36,338 | 46,837 |
| 3 | 93,852 | 53,019 | 52,978 |
| 4 | 63,575 | 53,139 | 52,448 |
| 5 | 53,992 | 53,474 | 44,852 |
| 6 | 52,265 | 54,757 | 42,306 |
| 7 | 47,439 | 70,952 | |



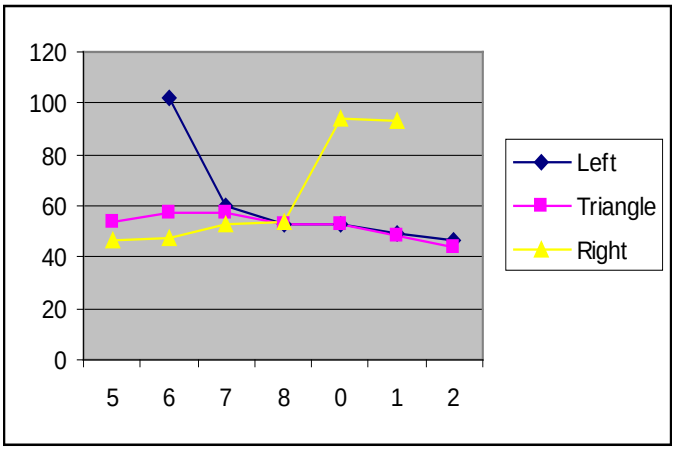

Edge 3-2

| | Left | Triangle | Right |
|---|---|---|---|
| 4 | | 44,01 | 46,837 |
| 5 | 54,128 | 45,953 | 46,837 |
| 6 | 67,684 | 47,258 | 47,075 |
| 7 | 57,902 | 52,492 | 50,485 |
| 8 | 52,978 | 53,385 | 38,5 |
| 0 | 53,019 | 93,416 | 96,077 |
| 1 | 49,043 | 92,925 | |




Edge 4-3

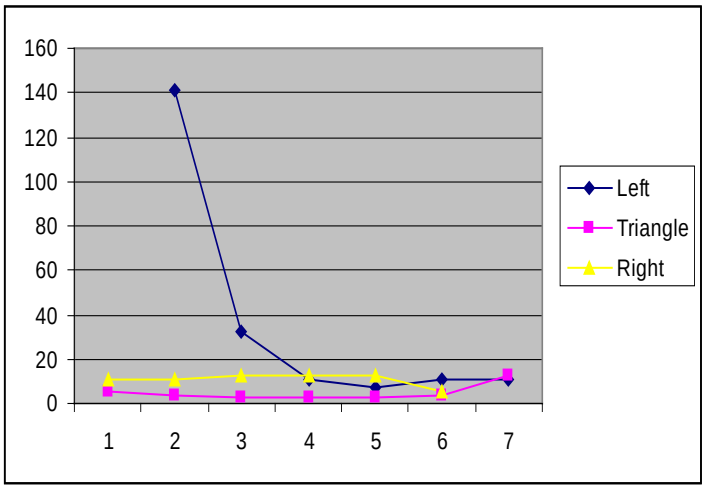
| | Left | Triangle | Right |
|---|---|---|---|
| 5 | | 54,128 | 46,837 |
| 6 | 102,119 | 57,29 | 47,258 |
| 7 | 60,35 | 57,15 | 52,492 |
| 8 | 52,448 | 52,978 | 53,385 |
| 0 | 53,139 | 53,147 | 93,852 |
| 1 | 49,043 | 48,456 | 92,925 |
| 2 | 46,837 | 44,01 | |



---Min_Max Radii-Ratio

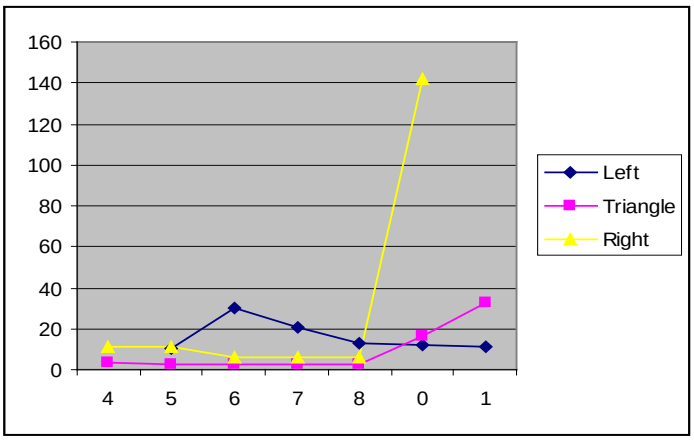
Edge 0-8

| | Left | Triangle | Right |
|---|---|---|---|
| 1 | | 5,153 | 10,981 |
| 2 | 141,516 | 3,331 | 10,981 |
| 3 | 32,632 | 2,579 | 12,803 |
| 4 | 10,493 | 2,657 | 12,803 |
| 5 | 7,603 | 3,126 | 12,803 |
| 6 | 10,981 | 3,473 | 5,361 |
| 7 | 10,981 | 12,699 | |



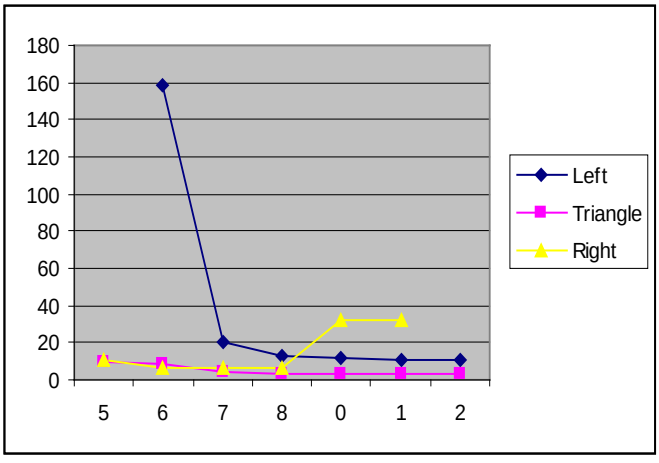

Edge 3-2

| | Left | Triangle | Right |
|---|---|---|---|
| 4 | | 3,096 | 10,981 |
| 5 | 10,109 | 2,312 | 10,981 |
| 6 | 29,816 | 2,233 | 6,332 |
| 7 | 20,28 | 2,307 | 6,332 |
| 8 | 12,803 | 2,772 | 6,332 |
| 0 | 11,727 | 16,324 | 141,516 |
| 1 | 11,083 | 32,632 | |




Edge 4-3

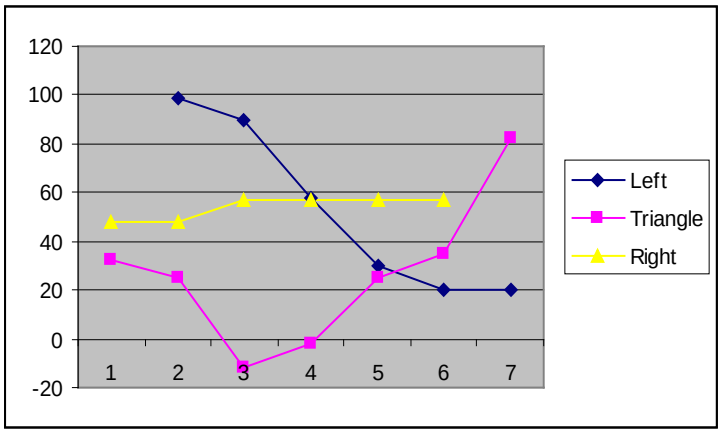
| | Left | Triangle | Right |
|---|---|---|---|
| 5 | | 10,109 | 10,981 |
| 6 | 158,85 | 8,596 | 6,332 |
| 7 | 20,28 | 4,233 | 6,332 |
| 8 | 12,803 | 3,297 | 6,332 |
| 0 | 11,727 | 3,204 | 32,632 |
| 1 | 11,083 | 3,114 | 32,632 |
| 2 | 10,981 | 3,096 | |



---Min_Max Γωνία

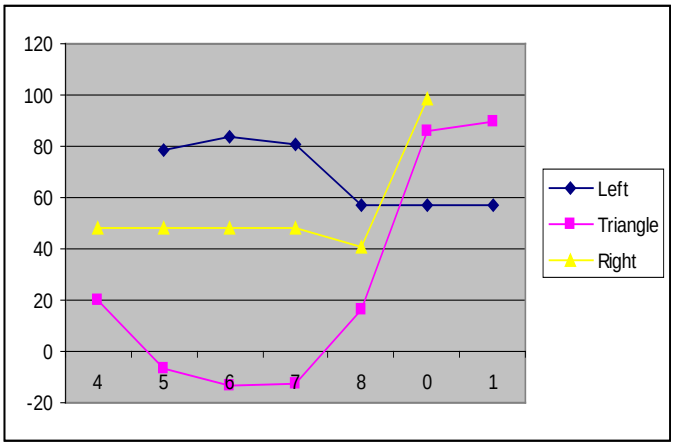
Edge 0-8

| | Left | Triangle | Right |
|---|---|---|---|
| 1 | | 32,654 | 48,273 |
| 2 | 98,584 | 24,762 | 48,273 |
| 3 | 89,755 | -11,883 | 57,362 |
| 4 | 57,722 | -1,893 | 57,362 |
| 5 | 29,734 | 25,142 | 57,362 |
| 6 | 20,24 | 35,038 | 57,362 |
| 7 | 19,762 | 82,102 | |



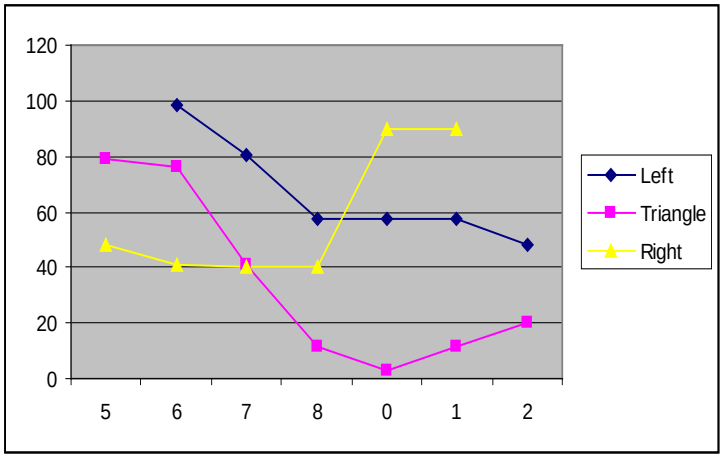

Edge 3-2

| | Left | Triangle | Right |
|---|---|---|---|
| 4 | | 19,762 | 48,273 |
| 5 | 78,757 | -6,542 | 48,273 |
| 6 | 83,424 | -13,38 | 48,273 |
| 7 | 80,678 | -12,799 | 48,273 |
| 8 | 57,362 | 16,617 | 40,467 |
| 0 | 57,362 | 85,632 | 98,584 |
| 1 | 57,362 | 89,755 | |




Edge 4-3

| | Left | Triangle | Right |
|---|---|---|---|
| 5 | | 78,757 | 48,273 |
| 6 | 98,303 | 75,87 | 41,083 |
| 7 | 80,678 | 41,253 | 40,467 |
| 8 | 57,362 | 11,224 | 40,467 |
| 0 | 57,362 | 2,557 | 89,755 |
| 1 | 57,362 | 11,346 | 89,755 |
| 2 | 48,273 | 19,762 | |

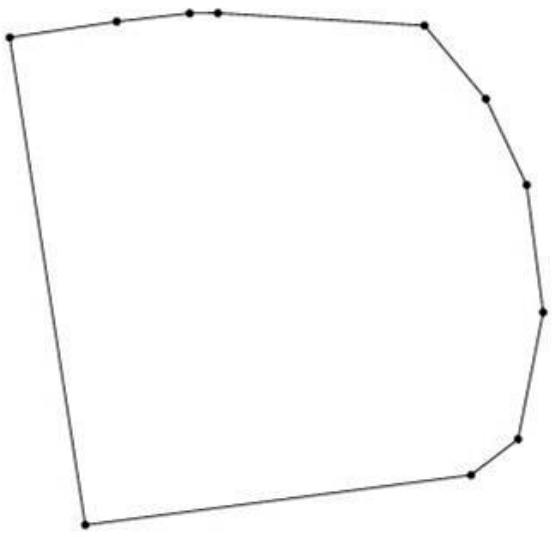


*4.4.2. Πολύγωνο Poly 2*

Number of vertices: 11

Coordinates of vertices

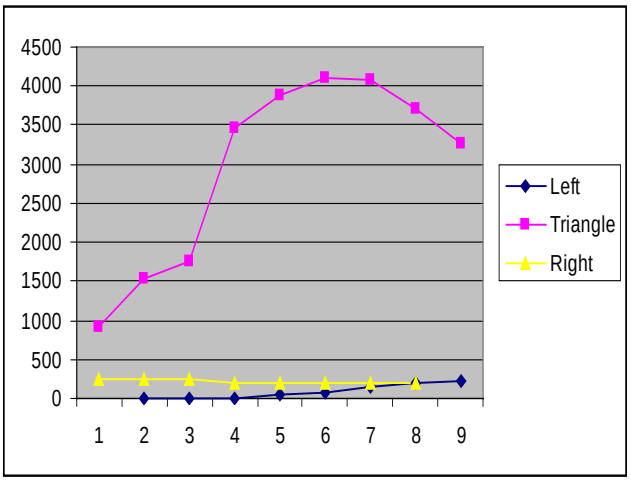
| | | |
|---|---|---|
| 0: | -49.45896 | 43.17748 |
| 1: | -29.68132 | 46.11089 |
| 2: | -16.16305 | 47.62734 |
| 3: | -10.96878 | 47.67484 |
| 4: | 27.22945 | 45.37130 |
| 5: | 38.59296 | 31.84319 |
| 6: | 46.13649 | 15.97604 |
| 7: | 49.19334 | -7.51068 |
| 8: | 44.53913 | -30.90217 |
| 9: | 35.85783 | -37.51687 |
| 10: | -35.49515 | -46.72448 |



---Max_Min εμβαδόν

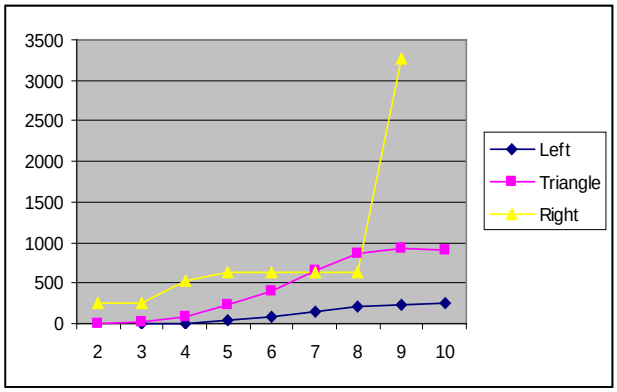
Edge 0-10

| | Left | Triangle | Right |
|---|---|---|---|
| 1 | | 909,505 | 244,585 |
| 2 | 4,831 | 1527,752 | 244,585 |
| 3 | 4,831 | 1761,571 | 245,287 |
| 4 | 6,89 | 3462,536 | 196,022 |
| 5 | 42,294 | 3878,885 | 196,022 |
| 6 | 83,682 | 4107,191 | 196,022 |
| 7 | 144,753 | 4080,618 | 196,022 |
| 8 | 205,393 | 3708,089 | 196,022 |
| 9 | 222,366 | 3271,673 | |



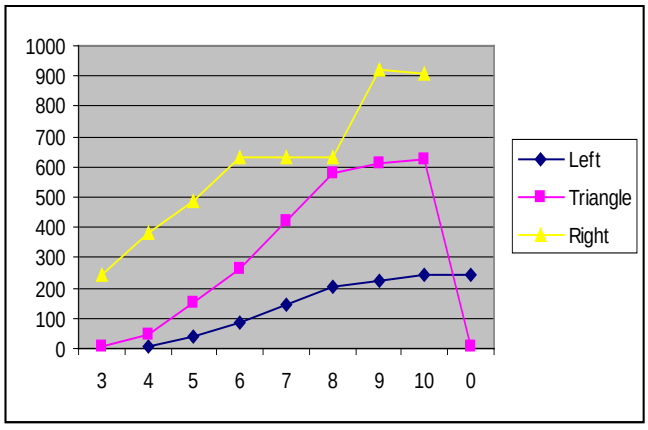

Edge 1-0

| | Left | Triangle | Right |
|---|---|---|---|
| 2 | | 4,831 | 244,585 |
| 3 | 3,617 | 11,98 | 245,287 |
| 4 | 6,89 | 90,785 | 531,189 |
| 5 | 42,294 | 241,229 | 632,438 |
| 6 | 83,682 | 409,201 | 632,438 |
| 7 | 144,753 | 645,94 | 632,438 |
| 8 | 205,393 | 870,428 | 632,438 |
| 9 | 222,366 | 923,107 | 3271,673 |
| 10 | 244,585 | 909,505 | |




Edge 2-1

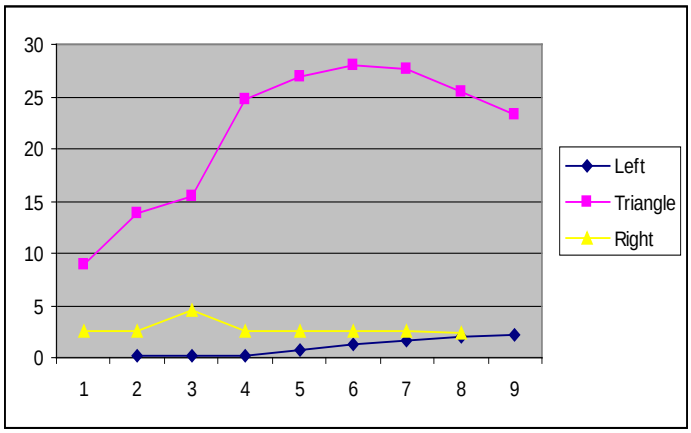
| | Left | Triangle | Right |
|---|---|---|---|
| 3 | | 3,617 | 245,287 |
| 4 | 6,89 | 48,15 | 380,745 |
| 5 | 42,294 | 148,205 | 487,845 |
| 6 | 83,682 | 261,173 | 632,438 |
| 7 | 144,753 | 422,24 | 632,438 |
| 8 | 205,393 | 576,818 | 632,438 |
| 9 | 222,366 | 614,945 | 923,107 |
| 10 | 244,585 | 623,078 | 909,505 |
| 0 | 244,585 | 4,831 | |



---Max_Min inradius

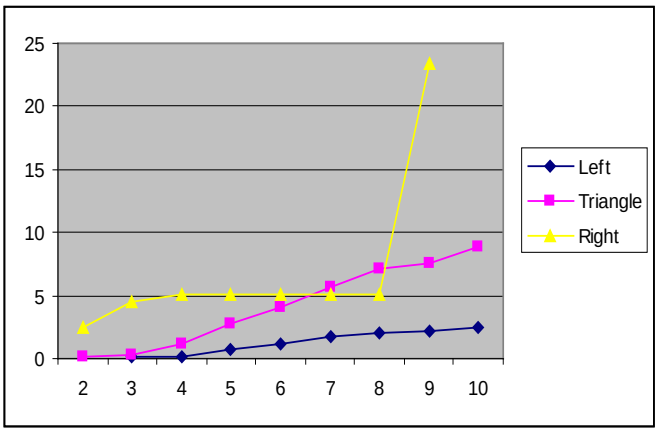
Edge 0-10

| | Left | Triangle | Right |
|---|---|---|---|
| 1 | | 8,917 | 2,458 |
| 2 | 0,144 | 13,833 | 2,458 |
| 3 | 0,193 | 15,502 | 4,544 |
| 4 | 0,193 | 24,81 | 2,549 |
| 5 | 0,741 | 26,96 | 2,549 |
| 6 | 1,192 | 28,007 | 2,549 |
| 7 | 1,68 | 27,645 | 2,549 |
| 8 | 2,047 | 25,377 | 2,384 |
| 9 | 2,2 | 23,339 | |



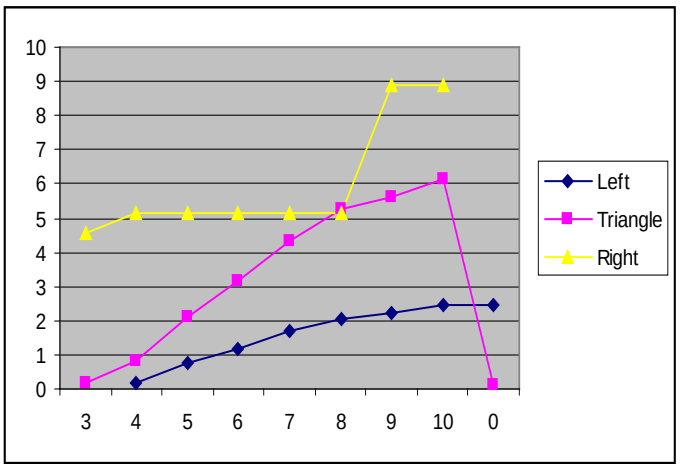

Edge 1-0

| | Left | Triangle | Right |
|---|---|---|---|
| 2 | | 0,144 | 2,458 |
| 3 | 0,193 | 0,309 | 4,544 |
| 4 | 0,193 | 1,182 | 5,1 |
| 5 | 0,741 | 2,703 | 5,1 |
| 6 | 1,192 | 4,072 | 5,1 |
| 7 | 1,68 | 5,709 | 5,1 |
| 8 | 2,047 | 7,059 | 5,1 |
| 9 | 2,2 | 7,576 | 23,339 |
| 10 | 2,458 | 8,917 | |




Edge 2-1

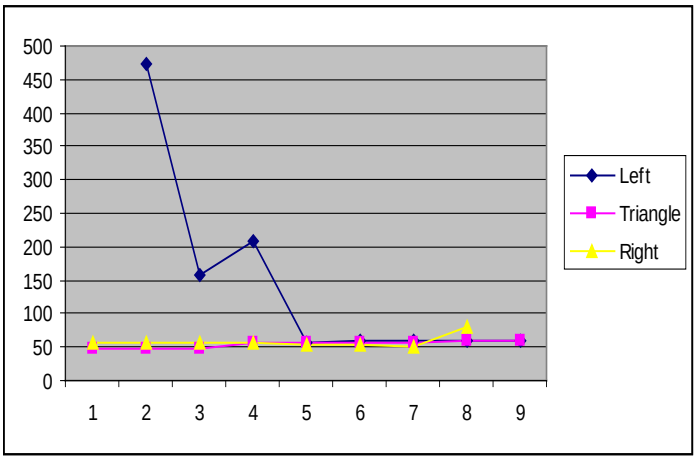
| | Left | Triangle | Right |
|---|---|---|---|
| 3 | | 0,193 | 4,544 |
| 4 | 0,159 | 0,845 | 5,174 |
| 5 | 0,741 | 2,112 | 5,174 |
| 6 | 1,192 | 3,164 | 5,174 |
| 7 | 1,68 | 4,342 | 5,174 |
| 8 | 2,047 | 5,248 | 5,174 |
| 9 | 2,2 | 5,6 | 8,917 |
| 10 | 2,458 | 6,141 | 8,917 |
| 0 | 2,458 | 0,144 | |



---Max_Min circumradius

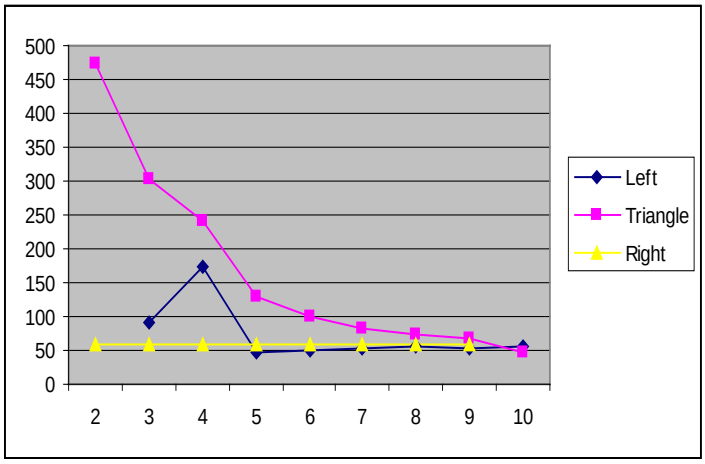
Edge 0-10

| | Left | Triangle | Right |
|---|---|---|---|
| 1 | | 46,51 | 55,735 |
| 2 | 472,764 | 48,167 | 55,735 |
| 3 | 157,019 | 48,802 | 55,735 |
| 4 | 208,758 | 56,155 | 55,735 |
| 5 | 56,634 | 56,217 | 54,27 |
| 6 | 58,043 | 56,655 | 52,849 |
| 7 | 59,368 | 57,696 | 50,485 |
| 8 | 60,991 | 59,891 | 81,701 |
| 9 | 60,635 | 58,736 | |



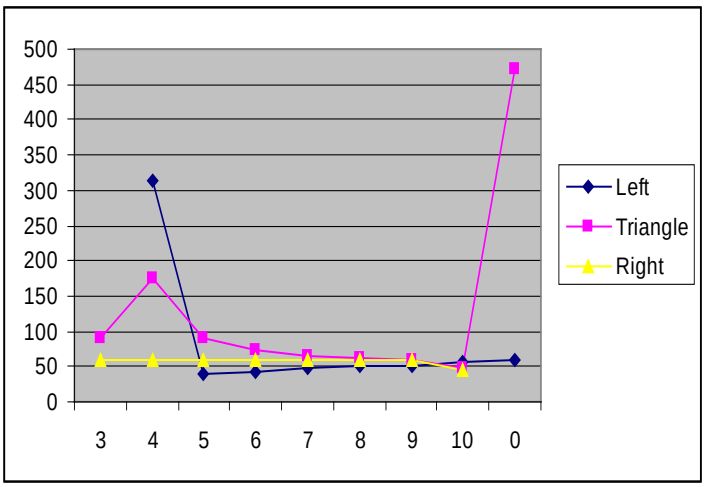

Edge 1-0

| | Left | Triangle | Right |
|---|---|---|---|
| 2 | | 472,764 | 59,891 |
| 3 | 91,7 | 303,612 | 59,891 |
| 4 | 174,666 | 240,417 | 59,891 |
| 5 | 46,052 | 128,309 | 59,891 |
| 6 | 48,916 | 99,053 | 59,891 |
| 7 | 51,922 | 81,859 | 59,891 |
| 8 | 55,005 | 73,509 | 59,891 |
| 9 | 53,483 | 67,563 | 58,736 |
| 10 | 55,735 | 46,51 | |




Edge 2-1

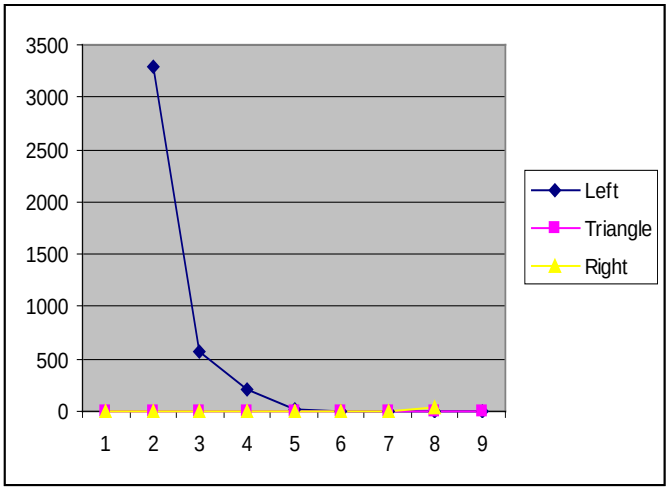
| | Left | Triangle | Right |
|---|---|---|---|
| 3 | | 91,7 | 59,891 |
| 4 | 313,407 | 174,666 | 59,891 |
| 5 | 38,963 | 91,205 | 59,891 |
| 6 | 43,038 | 74,236 | 59,891 |
| 7 | 47,381 | 65,684 | 59,891 |
| 8 | 51,567 | 62,589 | 59,891 |
| 9 | 49,89 | 58,628 | 58,736 |
| 10 | 55,735 | 48,896 | 46,51 |
| 0 | 59,891 | 472,764 | |



---Max_Min radii ratio

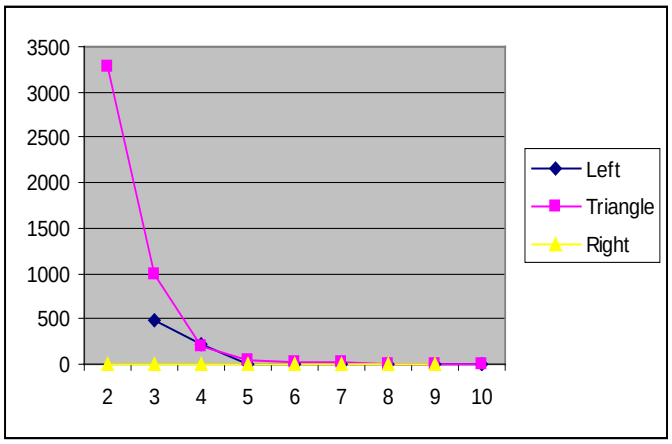
Edge 0-10

| | Left | Triangle | Right |
|---|---|---|---|
| 1 | | 5,216 | 3,833 |
| 2 | 3287,343 | 3,482 | 3,833 |
| 3 | 565,431 | 3,148 | 3,833 |
| 4 | 203,421 | 2,263 | 5,579 |
| 5 | 9,764 | 2,085 | 5,579 |
| 6 | 9,27 | 2,023 | 5,579 |
| 7 | 6,959 | 2,087 | 5,579 |
| 8 | 6,821 | 2,36 | 34,269 |
| 9 | 6,821 | 2,517 | |



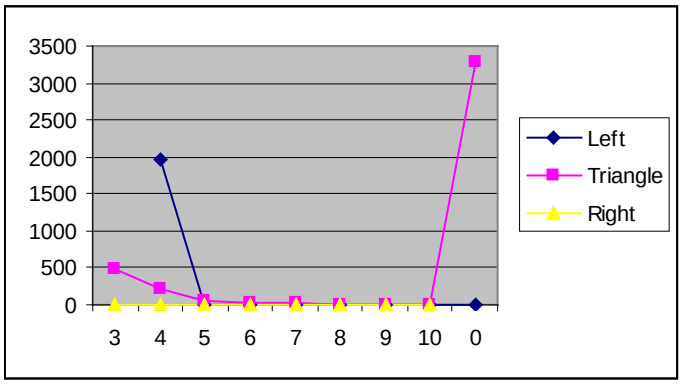

Edge 1-0

| | Left | Triangle | Right |
|---|---|---|---|
| 2 | | 3287,343 | 3,482 |
| 3 | 476,261 | 982,344 | 3,148 |
| 4 | 220,325 | 203,421 | 2,517 |
| 5 | 8,729 | 47,478 | 2,517 |
| 6 | 8,729 | 24,324 | 2,517 |
| 7 | 6,959 | 14,338 | 2,517 |
| 8 | 6,821 | 10,414 | 2,517 |
| 9 | 6,821 | 8,917 | 2,517 |
| 10 | 3,833 | 5,216 | |




Edge 2-1

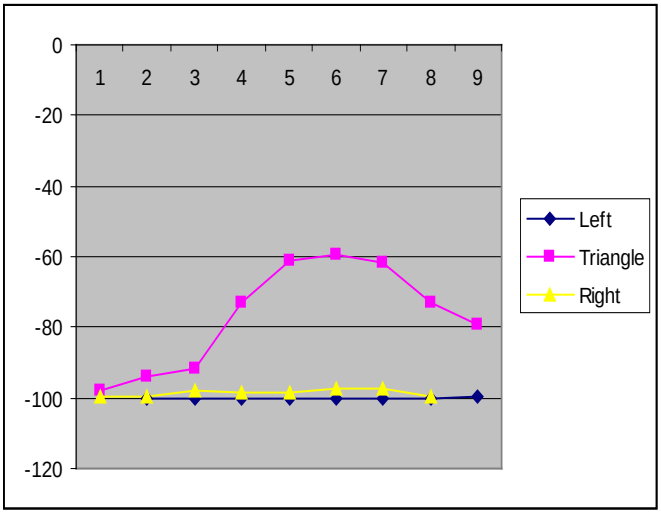
| | Left | Triangle | Right |
|---|---|---|---|
| 3 | | 476,261 | 3,833 |
| 4 | 1976,785 | 206,714 | 2,757 |
| 5 | 8,197 | 43,182 | 2,517 |
| 6 | 8,197 | 23,46 | 2,517 |
| 7 | 6,959 | 15,127 | 2,517 |
| 8 | 6,821 | 11,926 | 2,517 |
| 9 | 6,821 | 10,47 | 2,517 |
| 10 | 3,833 | 7,963 | 5,216 |
| 0 | 3,482 | 3287,343 | |



---Max_Min Γωνία

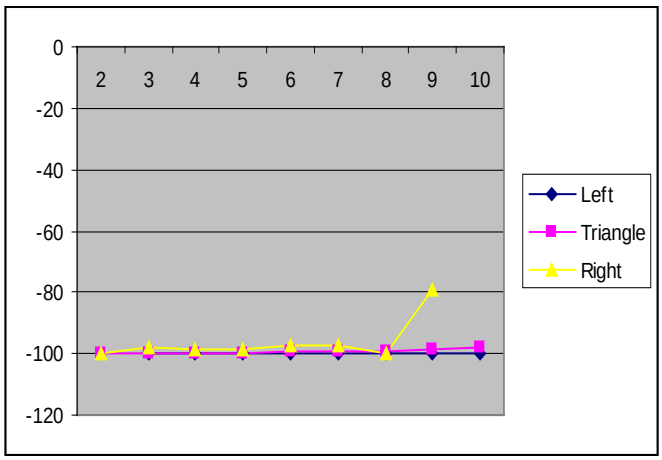
Edge 0-10

| | Left | Triangle | Right |
|---|---|---|---|
| 1 | | -97,663 | -99,864 |
| 2 | -99,99 | -93,723 | -99,864 |
| 3 | -99,96 | -91,78 | -98,055 |
| 4 | -99,96 | -73,032 | -98,752 |
| 5 | -99,959 | -61,362 | -98,727 |
| 6 | -99,933 | -59,607 | -97,55 |
| 7 | -99,914 | -61,511 | -97,55 |
| 8 | -99,907 | -73,219 | -99,777 |
| 9 | -99,895 | -79,052 | |



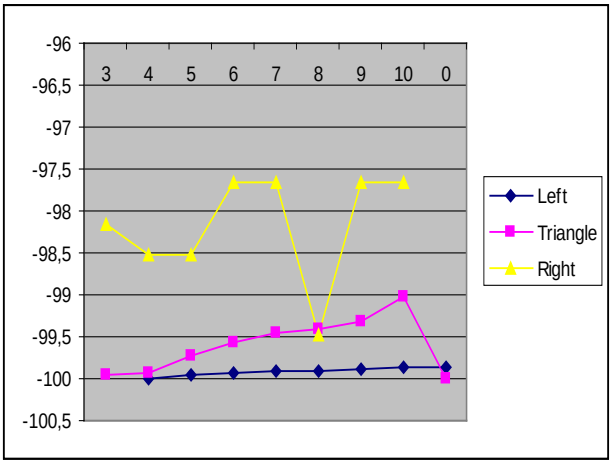

Edge 1-0

| | Left | Triangle | Right |
|---|---|---|---|
| 2 | | -99,99 | -99,864 |
| 3 | -99,96 | -99,952 | -98,055 |
| 4 | -99,96 | -99,914 | -98,752 |
| 5 | -99,959 | -99,696 | -98,727 |
| 6 | -99,933 | -99,489 | -97,55 |
| 7 | -99,914 | -99,251 | -97,55 |
| 8 | -99,907 | -99,071 | -99,594 |
| 9 | -99,895 | -98,899 | -79,052 |
| 10 | -99,864 | -97,663 | |




Edge 2-1

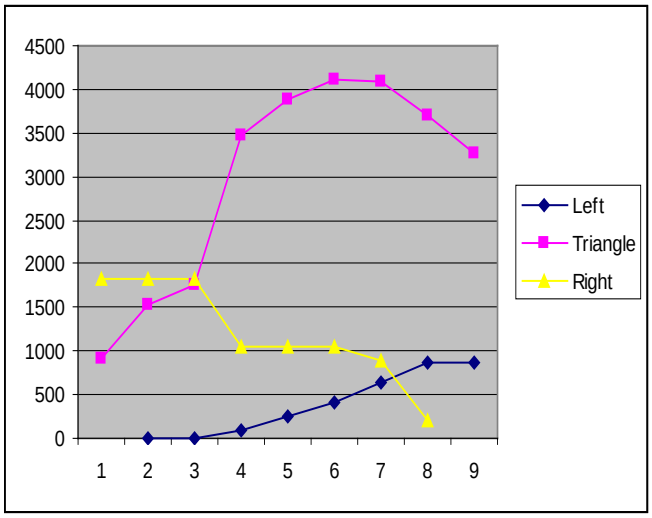
| | Left | Triangle | Right |
|---|---|---|---|
| 3 | | -99,96 | -98,169 |
| 4 | -99,997 | -99,924 | -98,519 |
| 5 | -99,959 | -99,722 | -98,519 |
| 6 | -99,933 | -99,579 | -97,663 |
| 7 | -99,914 | -99,462 | -97,663 |
| 8 | -99,907 | -99,408 | -99,478 |
| 9 | -99,895 | -99,325 | -97,663 |
| 10 | -99,864 | -99,028 | -97,663 |
| 0 | -99,864 | -99,99 | |



---Min_Max Εμβαδόν

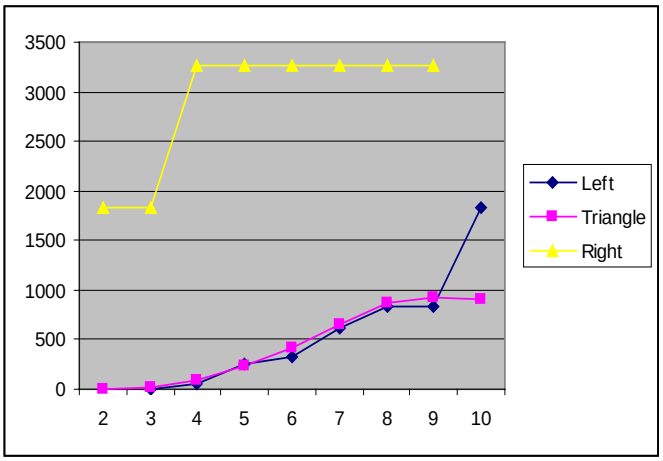
Edge 0-10

| | Left | Triangle | Right |
|---|---|---|---|
| 1 | | 909,505 | 1831,192 |
| 2 | 4,831 | 1527,752 | 1831,192 |
| 3 | 10,766 | 1761,571 | 1831,192 |
| 4 | 90,785 | 3462,536 | 1054,463 |
| 5 | 245,287 | 3878,885 | 1054,463 |
| 6 | 409,201 | 4107,191 | 1054,463 |
| 7 | 645,94 | 4080,618 | 899,24 |
| 8 | 870,428 | 3708,089 | 196,022 |
| 9 | 870,428 | 3271,673 | |



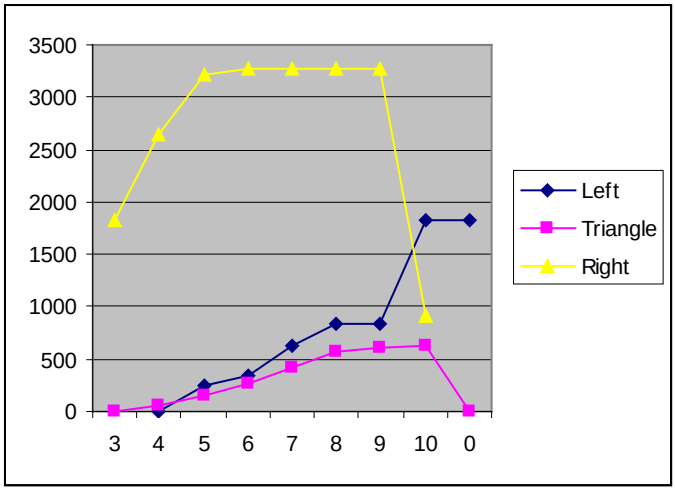

Edge 1-0

| | Left | Triangle | Right |
|---|---|---|---|
| 2 | | 4,831 | 1831,192 |
| 3 | 3,617 | 11,98 | 1831,192 |
| 4 | 48,15 | 90,785 | 3271,673 |
| 5 | 245,287 | 241,229 | 3271,673 |
| 6 | 333,489 | 409,201 | 3271,673 |
| 7 | 622,159 | 645,94 | 3271,673 |
| 8 | 832,063 | 870,428 | 3271,673 |
| 9 | 832,063 | 923,107 | 3271,673 |
| 10 | 1831,192 | 909,505 | |




Edge 2-1

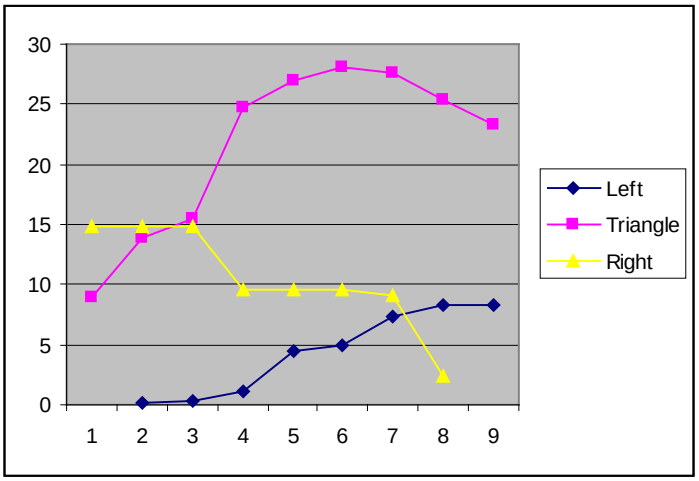
| | Left | Triangle | Right |
|---|---|---|---|
| 3 | | 3,617 | 1831,192 |
| 4 | 6,89 | 48,15 | 2643,816 |
| 5 | 245,287 | 148,205 | 3210,609 |
| 6 | 333,489 | 261,173 | 3271,673 |
| 7 | 622,159 | 422,24 | 3271,673 |
| 8 | 832,063 | 576,818 | 3271,673 |
| 9 | 832,063 | 614,945 | 3271,673 |
| 10 | 1831,192 | 623,078 | 909,505 |
| 0 | 1831,192 | 4,831 | |



---Min_Max inradius

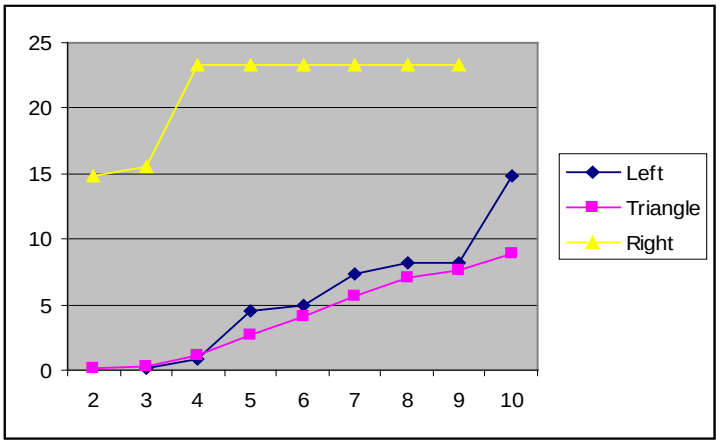
Edge 0-10

| | Left | Triangle | Right |
|---|---|---|---|
| 1 | | 8,917 | 14,814 |
| 2 | 0,144 | 13,833 | 14,814 |
| 3 | 0,278 | 15,502 | 14,814 |
| 4 | 1,182 | 24,81 | 9,588 |
| 5 | 4,544 | 26,96 | 9,588 |
| 6 | 4,944 | 28,007 | 9,588 |
| 7 | 7,292 | 27,645 | 9,049 |
| 8 | 8,251 | 25,377 | 2,384 |
| 9 | 8,251 | 23,339 | |



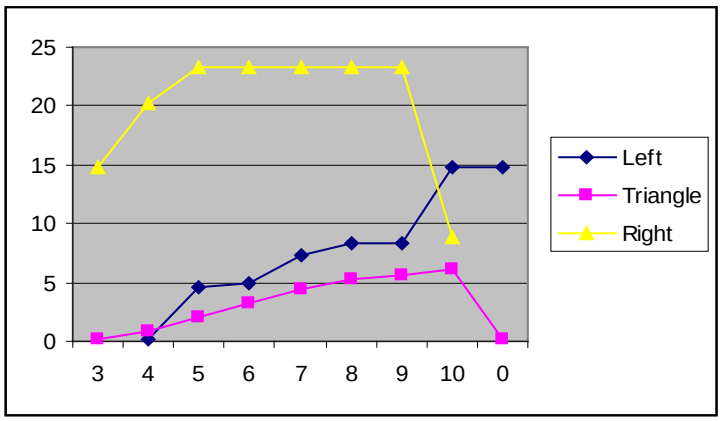

Edge 1-0

| | Left | Triangle | Right |
|---|---|---|---|
| 2 | | 0,144 | 14,814 |
| 3 | 0,193 | 0,309 | 15,502 |
| 4 | 0,845 | 1,182 | 23,339 |
| 5 | 4,544 | 2,703 | 23,339 |
| 6 | 4,944 | 4,072 | 23,339 |
| 7 | 7,292 | 5,709 | 23,339 |
| 8 | 8,251 | 7,059 | 23,339 |
| 9 | 8,251 | 7,576 | 23,339 |
| 10 | 14,814 | 8,917 | |




Edge 2-1

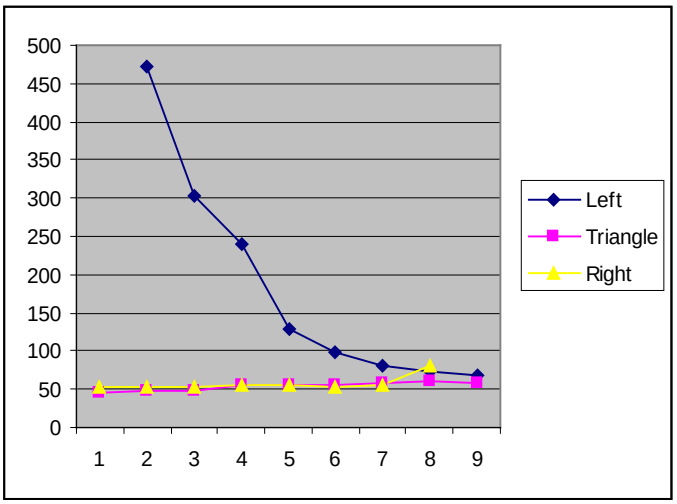
| | Left | Triangle | Right |
|---|---|---|---|
| 3 | | 0,193 | 14,814 |
| 4 | 0,159 | 0,845 | 20,231 |
| 5 | 4,544 | 2,112 | 23,339 |
| 6 | 4,944 | 3,164 | 23,339 |
| 7 | 7,292 | 4,342 | 23,339 |
| 8 | 8,251 | 5,248 | 23,339 |
| 9 | 8,251 | 5,6 | 23,339 |
| 10 | 14,814 | 6,141 | 8,917 |
| 0 | 14,814 | 0,144 | |



---Min_Max circumradius

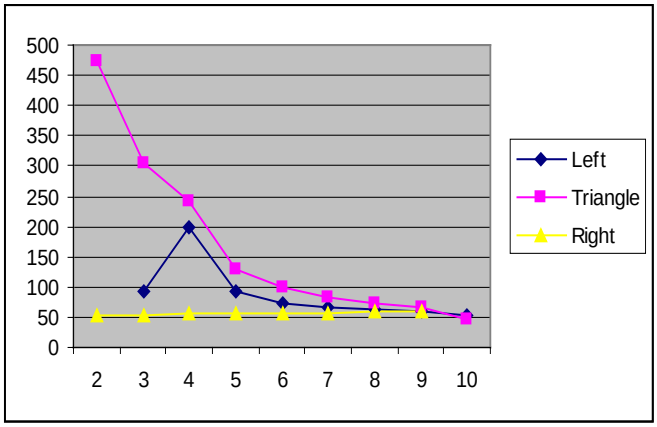
Edge 0-10

| | Left | Triangle | Right |
|---|---|---|---|
| 1 | | 46,51 | 52,393 |
| 2 | 472,764 | 48,167 | 52,393 |
| 3 | 303,612 | 48,802 | 52,393 |
| 4 | 240,417 | 56,155 | 56,091 |
| 5 | 128,309 | 56,217 | 55,222 |
| 6 | 99,053 | 56,655 | 54,186 |
| 7 | 81,859 | 57,696 | 54,62 |
| 8 | 73,509 | 59,891 | 81,701 |
| 9 | 67,563 | 58,736 | |



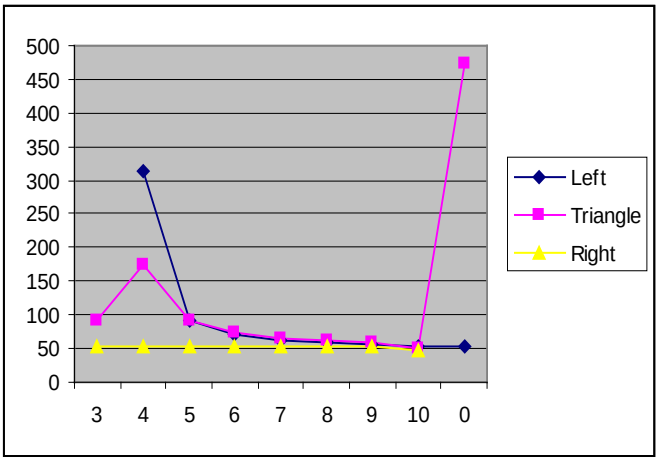

Edge 1-0

| | Left | Triangle | Right |
|---|---|---|---|
| 2 | | 472,764 | 52,393 |
| 3 | 91,7 | 303,612 | 52,393 |
| 4 | 198,835 | 240,417 | 56,155 |
| 5 | 91,205 | 128,309 | 56,217 |
| 6 | 74,236 | 99,053 | 56,655 |
| 7 | 65,684 | 81,859 | 57,696 |
| 8 | 62,589 | 73,509 | 60,635 |
| 9 | 58,628 | 67,563 | 58,736 |
| 10 | 52,393 | 46,51 | |




Edge 2-1

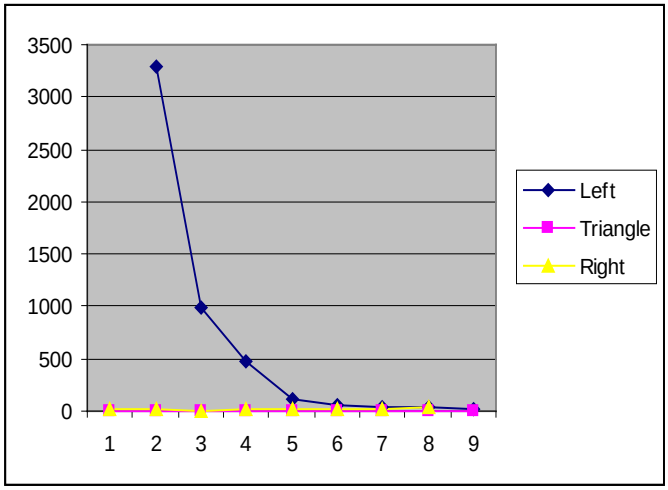
| | Left | Triangle | Right |
|---|---|---|---|
| 3 | | 91,7 | 52,393 |
| 4 | 313,407 | 174,666 | 54,107 |
| 5 | 91,036 | 91,205 | 54,107 |
| 6 | 70,827 | 74,236 | 54,107 |
| 7 | 62,627 | 65,684 | 54,107 |
| 8 | 60,374 | 62,589 | 54,107 |
| 9 | 56,647 | 58,628 | 54,107 |
| 10 | 52,393 | 48,896 | 46,51 |
| 0 | 52,393 | 472,764 | |



---Min_Max radii ratio

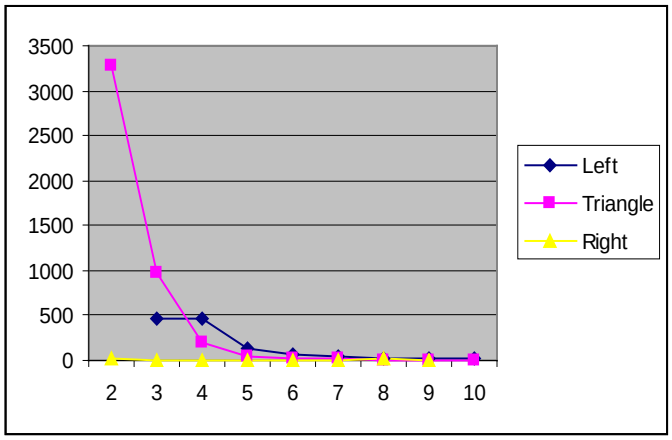
Edge 0-10

| | Left | Triangle | Right |
|---|---|---|---|
| 1 | | 5,216 | 20,294 |
| 2 | 3287,343 | 3,482 | 20,294 |
| 3 | 982,344 | 3,148 | 9,471 |
| 4 | 476,261 | 2,263 | 9,733 |
| 5 | 122,915 | 2,085 | 9,733 |
| 6 | 59,41 | 2,023 | 9,733 |
| 7 | 37,281 | 2,087 | 9,733 |
| 8 | 29,491 | 2,36 | 34,269 |
| 9 | 25,753 | 2,517 | |



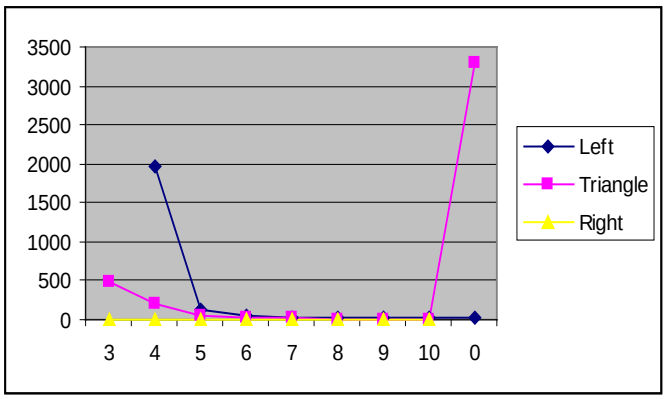

Edge 1-0

| | Left | Triangle | Right |
|---|---|---|---|
| 2 | | 3287,343 | 20,294 |
| 3 | 476,261 | 982,344 | 9,471 |
| 4 | 476,261 | 203,421 | 9,733 |
| 5 | 122,915 | 47,478 | 9,733 |
| 6 | 59,41 | 24,324 | 9,733 |
| 7 | 37,281 | 14,338 | 9,733 |
| 8 | 29,491 | 10,414 | 11,89 |
| 9 | 25,753 | 8,917 | 2,517 |
| 10 | 20,294 | 5,216 | |




Edge 2-1

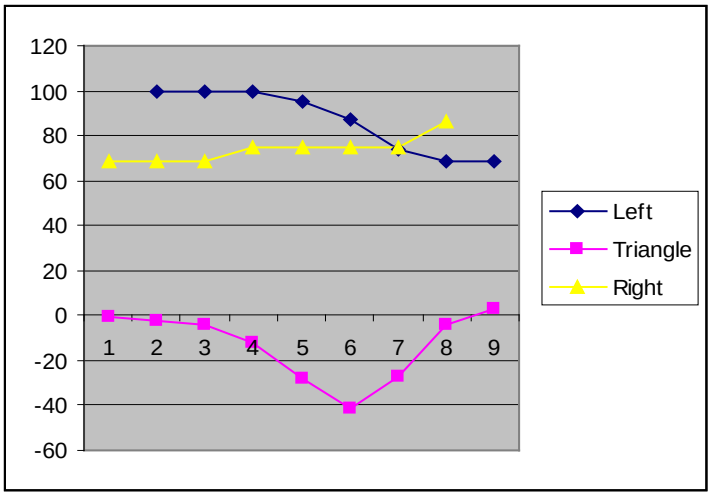
| | Left | Triangle | Right |
|---|---|---|---|
| 3 | | 476,261 | 9,471 |
| 4 | 1976,785 | 206,714 | 9,733 |
| 5 | 122,915 | 43,182 | 9,733 |
| 6 | 59,41 | 23,46 | 9,733 |
| 7 | 37,281 | 15,127 | 9,733 |
| 8 | 29,491 | 11,926 | 10,338 |
| 9 | 25,753 | 10,47 | 5,216 |
| 10 | 20,294 | 7,963 | 5,216 |
| 0 | 20,294 | 3287,343 | |



---Min_Max Γωνία

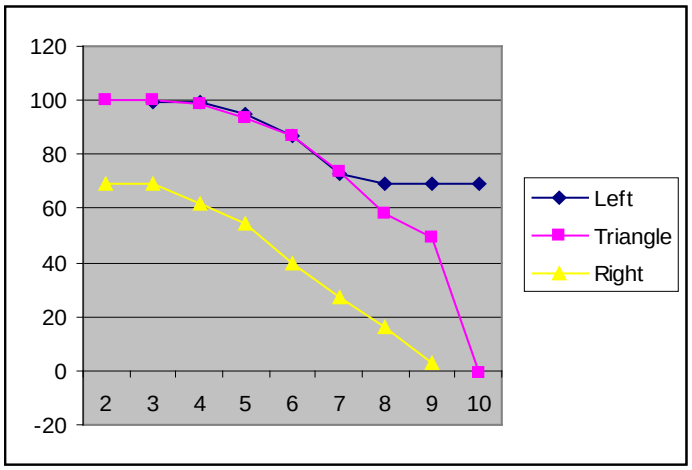
Edge 0-10

| | Left | Triangle | Right |
|---|---|---|---|
| 1 | | -0,685 | 68,811 |
| 2 | 99,937 | -2,123 | 68,811 |
| 3 | 99,796 | -3,777 | 68,811 |
| 4 | 99,474 | -12,516 | 74,963 |
| 5 | 94,976 | -27,838 | 74,963 |
| 6 | 86,985 | -41,806 | 74,963 |
| 7 | 73,556 | -27,592 | 74,963 |
| 8 | 68,811 | -4,107 | 86,644 |
| 9 | 68,811 | 2,575 | |



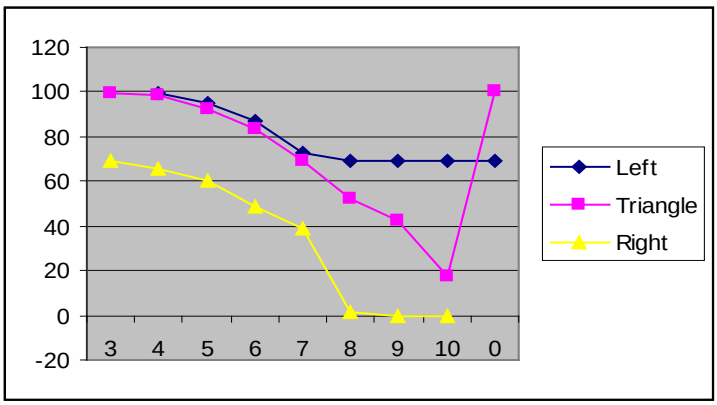

Edge 1-0

| | Left | Triangle | Right |
|---|---|---|---|
| 2 | | 99,937 | 68,811 |
| 3 | 99,474 | 99,796 | 68,811 |
| 4 | 99,474 | 98,719 | 62,103 |
| 5 | 94,976 | 93,825 | 54,115 |
| 6 | 86,985 | 86,504 | 39,553 |
| 7 | 73,072 | 73,556 | 27,465 |
| 8 | 68,811 | 58,078 | 16,142 |
| 9 | 68,811 | 49,469 | 2,575 |
| 10 | 68,811 | -0,685 | |




Edge 2-1

| | Left | Triangle | Right |
|---|---|---|---|
| 3 | | 99,474 | 68,811 |
| 4 | 99,759 | 98,664 | 65,308 |
| 5 | 94,976 | 92,401 | 60,503 |
| 6 | 86,985 | 83,549 | 48,621 |
| 7 | 73,072 | 68,768 | 39,002 |
| 8 | 68,811 | 51,956 | 1,362 |
| 9 | 68,811 | 42,299 | -0,685 |
| 10 | 68,811 | 17,337 | -0,685 |
| 0 | 68,811 | 99,937 | |



# ΚΕΦΑΛΑΙΟ 5. ΣΥΜΠΕΡΑΣΜΑΤΑ - ΕΠΕΚΤΑΣΕΙΣ

Μελετήθηκε η συμπεριφορά του αλγορίθμου για πολύγωνα λιγοστών κορυφών (έως και 13 κορυφών) για την περίπτωση μεγιστοποίησης της ελάχιστης τιμής του κριτηρίου και ελαχιστοποίησης της μέγιστης τιμής του κριτηρίου με κριτήρια το εμβαδόν των σχηματισμένων τριγώνων, την ακτίνα του εγγεγραμμένου και του περιγεγραμμένου κύκλου του τριγώνου καθώς και τη γωνία τριγώνου. Ο αλγόριθμος, που στηρίχθηκε στην τεχνική του δυναμικού προγραμματισμού έχει πολυπλοκότητα χρόνου $O(n^3)$ και πολυπλοκότητα χώρου $O(n^2)$. Δεν αποτελεί βέλτιστο αλγόριθμο χρονικά για τα κριτήρια του εμβαδού και της γωνίας, μιας και για την περίπτωση του εμβαδού, το πρόβλημα έχει επιλυθεί από τους *Keil* & *Vassilev* σε χρόνο $O(n^2 \log n)$ [6] , ενώ για την περίπτωση της Max_Min γωνίας, η *Delaunay* διαίρεση σε τρίγωνα δίνει βέλτιστη λύση σε $O(n \log n)$ χρόνο [9], όπως έχει αναφερθεί. Και για την περίπτωση της Min_Max γωνίας υπάρχει βέλτιστος αλγόριθμος που χρησιμοποιεί την τεχνική εισαγωγής ακμής και επιλύει το πρόβλημα σε χρόνο $O(n^2 \log n)$ [8].

Η μελέτη του αλγορίθμου σε μεγαλύτερου μεγέθους πολύγωνα για μια πιο εμπεριστατωμένη μελέτη θα μπορούσε να είναι μια επέκταση της εργασίας και η εξαγωγή συμπερασμάτων. Όπως επίσης θα μπορούσε να μελετηθεί η επέκταση του αλγορίθμου σε μη κυρτά πολύγωνα.

Για την επέκταση του αλγορίθμου σε μη κυρτά πολύγωνα αρκεί για την επίλυση ενός υποπροβλήματος που ορίζεται από τις κορυφές i και j, να ληφθούν υπόψιν μόνο οι κορυφές που ορίζουν διαγωνίους με τις i και j. Σε αυτή την περίπτωση χρειάζεται ένας πίνακας, ο οποίος για ζεύγος κορυφών θα καταχωρεί έναν αριθμό, ο οποίος θα δηλώνει εάν αυτές οι κορυφές ορίζουν διαγώνιο ή όχι.

# ΑΝΑΦΟΡΕΣ

[1] E.F.D'Azevedoand, R.B.Simpson,"Optimal interpolation triangle incidences", Siam J. Sci. Stat. Comput. 10, pp 1063-1075, 1989

[2] E.L. Lloyd, «On triangulations of a set of points in the plane", In Proc. 18th Ann. IEEE Sympos. Found. Comput. Sci, pp 228-240.

[3] E. P. Preparata and M.I. Shamos, Computational Geometry, Springer-Verlang, New York 1985.

[4] H. Edelsbrunner and T. S. Tan, "A quadratic time algorithm for the minmax triangulation", In Proc. 32nd IEEE Sympos. Found. Comput. Sci., pp 414-423, 1991.

[5] H. Edelsbrunner, T. S. Tan and R. Wauotitsch, "An O ($n^2$logn) time algorithm for the minmax angle triangulation", Siam J. Comput. 13, pp 994-1008, 1992.

[6] J.M. Keil, T.S. Vassilev, Computational Geometry 35, pp 173-187, 2006.

[7] Paul Chew, Constrained Delaunay Triangulations, Algorithmica 4, pp 97-108, 1989.

[8] M. Bern, H. Edelsbrunner, D. Eppstein, S. Mitchell και T.S. Tan, Edge insertion for optimal triangulations, Discrete and Computational Geometry 10, pp 47-65, 1993.

[9] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf, "Computational Geometry: Algorithms and Applications", Second Edition, Springer

[10] O. Aichholzer, F. Hurtado, and M. Noy. A lower bound on the number of triangulations of planar point sets. Computational Geometry: Theory and Applications, 29(2) pp 135–145, 2004.

[11] Sibson,"Locally equiangular triangulations", The comput. J.21,1978.

[12] T.J. Baker, P.P. Pebay, A comparison of triangle quality measures, in: Proceedings of the 10th International Meshing Roundtable, pp.327-340, October 2001.

# ΠΑΡΑΡΤΗΜΑ

# ΣΥΝΤΟΜΟ ΒΙΟΓΡΑΦΙΚΟ

Άγιος Ιωάννης Πασσαρώνας Τηλέφωνο 2651-0-63017
E-mail lkamona@cs.uoi..gr

# Λαμπρινή Καμωνά

**Προσωπικές πληροφορίες**

- Οικογενειακή κατάσταση: Άγαμη
- Ηλικία:28
- Τόπος γέννησης:Ιωάννινα

**Σπουδές**

[ 1997-2001 ] Τμήμα Πληροφορικής Πανεπιστήμιο Ιωαννίνων
**6,76**

**Γλώσσες**

Αγγλική, First Certificate

**Εργασία**

[ 2002- Σήμερα ] Εκπαιδευτικός Δευτεροβάθμιας Εκπαίδευσης
Νομού Ιωαννίνων