Υλοποίηση Αλγορίθμων για τον Υπολογισμό του Μικρότερου Τριγώνου Περιγεγραμμένου σε Δοθέν Κυρτό Πολύγωνο

Ευάγγελος Παππάς

Διπλωματική Εργασία

Επιβλέπων: Λ. Παληός

Ιωάννινα, Φεβρουάριος, 2024

ΤΜΗΜΑ ΜΗΧ. Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF IOANNINA

Περίληψη

Το πρόβλημα των περιγεγραμμένων και εγγράφων τριγώνων σε δοθέν κυρτό πολύγωνο, με ελάχιστο και μέγιστο αντίστοιχα εμβαδό έχει μελετηθεί εκτενώς λόγω της εφαρμογής του στη ρομποτική και στα προβλήματα ανίχνευση σύγκρουσης (collision detection). Στις εφαρμογές αυτές είναι ευκολότερο να ελεγχθούν αντικείμενα με όσο το δυνατόν λιγότερες πλευρές, στόχο τον οποίο προσπαθεί να επιτύχει η παρούσα διπλωματική εργασία. Συγκεκριμένα, υλοποιήθηκαν δύο αλγόριθμοι. Ο πρώτος ονομάζεται «Minimal Enclosing Triangles» και έχει ως στόχο να υπολογίσει τα τοπικά ελάχιστα τρίγωνα που περικλείουν ένα κυρτό πολύγωνο και κατ' επέκταση το ολικό ελάχιστο. Ο δεύτερος, είναι ένας ακολουθιακός αλγόριθμος, ο οποίος αξιοποιεί το συμπέρασμα του πρώτου, με σκοπό την εύρεση του μέγιστου δυνατού τριγώνου το οποίο είναι εγγεγραμμένο στο κυρτό πολύγωνο. Ως ελάχιστο ή μέγιστο χαρακτηρίζουμε ένα τρίγωνο με κριτήριο το εμβαδόν του. Τέλος, για τη βαθύτερη κατανόηση των αλγορίθμων, υλοποιήθηκε η οπτικοποίηση του τρόπου εκτέλεσης και των αποτελεσμάτων τους. Οι αλγόριθμοι υλοποιήθηκαν χρησιμοποιώντας ως γλώσσα προγραμματισμού την Python. Απαραίτητη προϋπόθεση για τη λειτουργεία του κώδικα, είναι η βιβλιοθήκη Matplotlib η οποία είναι υπεύθυνη για την οπτικοποίηση, καθώς επίσης και από τη βιβλιοθήκη Math τα sqrt, cos, sin, asin, degrees και pi τα οποία αναλαμβάνουν τα μαθηματικά.

Ολοκληρώνοντας, θα ήθελα να ευχαριστήσω τον καθηγητή μου κ. Λεωνίδα Παληό για την πολύτιμη βοήθειά του καθ' όλη τη διάρκεια εκπόνησης της εργασίας, καθώς και την οικογένειά μου για τη στήριξή της.

Λέξεις Κλειδιά: Ελάχιστο περιγεγραμμένο τρίγωνο, Μέγιστο εγγεγραμμένο τρίγωνο, Ακολουθιακός αλγόριθμος, οπτικοποίηση αλγορίθμου, τοπικό ελάχιστο, ολικό ελάχιστο, κυρτό πολύγωνο.

Πίνακας περιεχομένων

1

Εισαγωγή

Κύριο αντικείμενο της διπλωματικής εργασίας είναι η ανάπτυξη δύο αλγορίθμων όπως επίσης και η οπτικοποίηση αυτών. Σκοπός της οπτικοποίησης είναι η ευκολότερη κατανόηση των αλγορίθμων, καθώς παρουσιάζονται τα βήματα και οι διαδικασίες για την ολοκλήρωσή τους.

1.1 Βασικοί Ορισμοί

Προτού ξεκινήσουμε θα ήταν χρήσιμο να αναφερθούμε σε κάποιους σημαντικούς ορισμούς οι οποίοι θα χρησιμοποιηθούν στη συνέχεια.

Πολύγωνο: ορίζεται ένα επίπεδο σχήμα που αποτελείται από μη τεμνόμενα ευθύγραμμα τμήματα, τα οποία ενώνονται ανά ζεύγη για να σχηματίσουν μια κλειστή διαδρομή. Τα ευθύγραμμα τμήματα, τα οποία ονομάζονται ακμές, συναντώνται μόνο στα άκρα τους, που ονομάζονται κορυφές. Όταν όλες οι πλευρές και οι εσωτερικές γωνίες του πολύγωνου είναι ίσες, τότε λέγεται κανονικό πολύγωνο.

Κυρτό Πολύγωνο: ορίζεται ως ένα πολύγωνο με όλες τις εσωτερικές του γωνίες μικρότερες από 180^{o} και έχει τις εξής ιδιότητες:

- Όλες οι κορυφές του πολυγώνου δείχνουν προς τα έξω, μακριά από το εσωτερικό του σχήματος.
- Περιέχεται εξ ολοκλήρου σε ένα κλειστό ημιεπίπεδο που ορίζεται από κάθε μία από τις ακμές του.
- Το άθροισμα των εξωτερικών γωνιών κάθε κυρτού πολυγώνου είναι ίσο με 360^{o}.
- Οποιαδήποτε ευθεία, τέμνει το πολύγωνο το πολύ σε 2 σημεία.
- Όλες οι διαγώνιοι του πολυγώνου βρίσκονται εξ' ολοκλήρου στο εσωτερικό του.

Περιγεγραμμένο Τρίγωνο: ορίζεται το τρίγωνο το οποίο είναι εξωτερικό του πολυγώνου και το πολύγωνο περιέχεται εξ ολοκλήρου στο εσωτερικό του.

Ελάχιστο Περιγεγραμμένο Τρίγωνο: για να θεωρηθεί ελάχιστο ένα περιγεγραμμένο τρίγωνο πρέπει να ισχύουν οι εξής ιδιότητες:

- Για κάθε κυρτό πολύγωνο με εμβαδόν Α υπάρχει ένα τρίγωνο το οποίο το περικλείει με εμβαδόν το πολύ 2Α.
- Τα μέσα των ακμών του τρίγωνου εφάπτονται με το πολύγωνο.
- Η μια τουλάχιστον ακμή του εφάπτεται στο πολύγωνο.

Εγγεγραμμένο Τρίγωνο: ορίζεται το τρίγωνο το οποίο είναι εσωτερικό του πολυγώνου.

Μέγιστο Εγγεγραμμένο Τρίγωνο: για να θεωρηθεί μέγιστο ένα εγγεγραμμένο τρίγωνο πρέπει να ισχύουν οι εξής ιδιότητες:

- Για κάθε κορυφή του τριγώνου, η ευθεία που περνάει από αυτή και παράλληλα από την απέναντι πλευρά του, περιέχει εξ' ολοκλήρου το πολύγωνο στο κλειστό ημιεπίπεδο που ορίζεται από αυτή και τη κορυφή.
- Οι κορυφές του εγγεγραμμένου τρίγωνου είναι υποσύνολο των κορυφών του πολυγώνου.

1.2 Αντικείμενο της Διπλωματικής Εργασίας

Το πρόβλημα των περιγεγραμμένων και εγγράφων τριγώνων σε δοθέν κυρτό πολύγωνο, με ελάχιστο και μέγιστο αντίστοιχα εμβαδό έχει μελετηθεί εκτενώς λόγω της εφαρμογής τους στη ρομποτική και στα προβλήματα ανίχνευση σύγκρουσης (collision detection). Στις εφαρμογές αυτές είναι ευκολότερο να ελεγχθούν αντικείμενα με όσο το δυνατόν λιγότερες πλευρές. Για παράδειγμα, στην ανίχνευση σύγκρουσης και ιδιαίτερα στις εφαρμογές που την απαιτούν σε πραγματικό χρόνο, η αποτελεσματικότητα είναι το κλειδί. Μειώνοντας τις πλευρές ενός αντικειμένου βελτιώνουμε τις επιδόσεις του κώδικα, καθώς μας δίνεται η δυνατότητα να ελέγξουμε διάφορα αντικείμενα γρηγορότερα αλλά και να διαχειριστούμε αντικείμενα με πιο συνθέτη γεωμετρία.

Αντικείμενο της συγκεκριμένης διπλωματικής εργασίας, είναι η μελέτη και η υλοποίηση δύο αλγορίθμων. Ο πρώτος εξ αυτών, είναι μια υλοποίηση των Joseph O' Rourke, Alok Aggarwal, Sanjeev Maddila και Michael Baldwin [1]. Είναι ένας αλγόριθμος που υπολογίζει το τρίγωνο με το μικρότερο εμβαδόν που είναι περιγεγραμμένο σε ένα δοθέν κυρτό πολύγωνο. Μάλιστα ο συγκεκριμένος είναι και η βέλτιστη δυνατή λύση στο πρόβλημα αυτό. Στο παρελθόν έχουνε υπάρξει και άλλες υλοποιήσεις, όπως αυτή των Klee και Laskowski [2] η οποία είναι και βάση του. Ο επόμενος είναι ένας ακολουθιακός αλγόριθμος, ο οποίος αξιοποιεί το συμπέρασμα του πρώτου, με σκοπό την εύρεση του μέγιστου δυνατού τριγώνου το οποίο είναι εγγεγραμμένο, αυτή τη φορά, σε κυρτό πολύγωνο.

Το τελευταίο κομμάτι της εργασίας είναι η οπτικοποίησή τους, με την οποία επιτυγχάνεται η βαθύτερη κατανόηση στο τρόπο λειτουργίας τους αφού είναι ευκολότερο να διακρίνουμε τόσο τα χαρακτηριστικά όσο και τα βήματά τους, καθιστώντας τους έτσι πιο προσιτούς και κατανοητούς. Ως αποτέλεσμα η οπτικοποίηση γίνεται ένα εργαλείο το οποίο μας βοηθάει στην επικοινωνία του αλγορίθμου, την αναγνώριση σφαλμάτων ακόμα και στη βελτιστοποίησή του.

Η αναλυτική περιγραφή καθώς επίσης και η υλοποίηση των αλγορίθμων περιγράφονται στο κεφάλαιο 3.

1.3 Σχετικά Ερευνητικά Αποτελέσματα

Μια πρώτη λύση του προβλήματος ήταν αυτή της Brute Force μεθόδου. Παρόλο που είναι ένας πολύ απλός τρόπος επίτευξης του στόχου, η πολυπλοκότητά του όσον αφορά το χρόνο, την καθιστά απαγορευτική για μεγάλο όγκο δεδομένων. Πιο συγκεκριμένα, αν έστω n οι κορυφές του πολυγώνου, η brute force προσέγγιση θα χρειαζόταν χρόνο ίσο με $O(n^3)$. Αναλυτικότερα θα ελεγχόντουσαν όλοι οι δυνατοί συνδυασμοί για τρεις κορυφές, για κάθε συνδυασμό θα υπολογιζόταν το εμβαδόν του τριγώνου που θα σχηματιζόταν από αυτές και από όλους τους συνδυασμούς θα κρατούσαμε το μικρότερο δυνατό. Όπως γίνεται αντιληπτό, ένας πιο αποδοτικός τρόπος έπρεπε να βρεθεί.

Συγκεκριμένα οι Klee και Laskowski παρουσίασαν έναν αλγόριθμο ο οποίος βρίσκει όλα τα τοπικά ελάχιστα τρίγωνα, τα οποία περικλείουν ένα κυρτό πολύγωνο σε χρόνο $O(nlog^2n)$ [2]. Το βασικό στοιχείο που διαφοροποιεί το συγκεκριμένο αλγόριθμο από τους προηγούμενους, είναι ο γεωμετρικός χαρακτηρισμός αυτών τον ελάχιστων τριγώνων, το οποίο έχει ως αποτέλεσμα να αποφεύγεται η brute force προσέγγιση η οποία δεδομένα είναι μια ιδιαίτερα κοστοβόρα διαδικασία τόσο χρονικά όσο και υπολογιστικά. Πιο αναλυτικά, αποδείξανε ότι παρόλο που υπάρχουν άπειρα τοπικά ελάχιστα, μπορούν να κατηγοριοποιηθούν, σε τόσες ομάδες όσες και οι κορυφές του πολυγώνου, ανάλογα με το εμβαδόν τους. Τα τρίγωνα υπολογίζονται σε χρόνο $O(nlog^2n)$ και χρειάζεται επιπλέον $O(n)$ για την επιλογή του ολικού ελαχίστου.

Ένα χρόνο αργότερα, οι Joseph O'Rourke et.al [1] και βασιζόμενοι στον αλγόριθμο «Rotating Calipers» του Toussaint [3], δημοσίευσαν τη δική τους υλοποίηση, η οποία επιτυγχάνει να ελαττώσει την πολυπλοκότητα όσον αφορά το χρόνο υπολογισμού, καταλήγοντας σε έναν αλγόριθμο γραμμικού χρόνου, ο οποίος αποδείχθηκε ότι είναι ο βέλτιστος στο να βρίσκει τα τοπικά ελάχιστα και κατ' επέκταση το ολικό ελάχιστο. Η γραμμικότητα αυτή επιτυγχάνεται, πηγαίνοντας από το ένα τρίγωνο αναφοράς στο επόμενο σειριακά. Απαραίτητη προϋπόθεση ήταν να δοθούν οι κορυφές του πολυγώνου σε ωρολογιακή φορά. Στο πλαίσιο όμως της διπλωματικής εργασίας προστέθηκε η ικανότητα διαχείρισης κορυφών δοσμένων σε αντίθετη φορά από αυτή του ρολογιού.

Όσον αφορά την εύρεση των μέγιστων εγγεγραμμένων τριγώνων, από τους πρώτους ακολουθιακούς αλγορίθμους ήταν αυτός των J. E. Boyce et al. [4], ο οποίος στη συνέχεια βελτιώθηκε από τους Aggarwal και Park [5]. Ο χρόνος εκτέλεσής του ήταν O(kn+ nlogn) όπου n το πλήθος των κορυφών του πολυγώνου. Παρόλο που οι ακολουθιακοί αλγόριθμοι δεν είναι συνολικά βέλτιστοι για τιμές k και n, η συγκεκριμένη περίπτωση είναι ίσως από τις μοναδικές όπου πετυχαίνει ακριβώς αυτό. Αναλυτικότερα, οι Dobkin και Snyder [6] παρουσίασαν έναν αλγόριθμο ο οποίος σε χρόνο O(n) υπολόγιζε το μέγιστο εμβαδόν τριγώνου εσωτερικά του κυρτού πολυγώνου. Και αυτός ο αλγόριθμος είναι βασισμένος στον «Rotating Calipers» του Toussaint [3], ο οποίος υπολογίζει ένα πεπερασμένο αριθμό τριγώνων και επιλέγει το κατάλληλο. Αργότερα σε μια δημοσίευση των Chandran και Mount [7] αποδείχθηκε ότι, τα δύο αυτά προβλήματα έχουνε αρκετά κοινά σημεία, ως αποτέλεσμα να μπορούν να υπολογιστούν σε χρόνο O(n). Συγκεκριμένα κατηγοριοποιήσανε τα τρίγωνα σε O(n) ομάδες τις οποίες ονόμασαν P-stable triangles.

1.4 Δομή της Διπλωματικής Εργασίας

Η διπλωματική αυτή εργασία αποτελείται από 4 κεφάλαια, τα οποία αναπτύσσουμε παρακάτω.

Στο δεύτερο κεφάλαιο δίνονται περιγραφικά οι αλγόριθμοι «Minimal Enclosing Triangles», ο οποίος έχει ως στόχο να υπολογίσει τα τοπικά ελάχιστα τρίγωνα που περικλείουν ένα κυρτό πολύγωνο και κατ' επέκταση το ολικό ελάχιστο και ο «Maximal Enclosed Triangles». Ο αλγόριθμος αυτός θα είναι ακολουθιακός και θα χρησιμοποιεί το αποτέλεσμα του «Minimal Enclosing Triangles», με στόχο την εύρεση των τοπικών μεγίστων, αυτή τη φορά, τριγώνων που είναι εγγεγραμμένα σε ένα κυρτό πολύγωνο και κατ' επέκταση το ολικό. Επίσης θα δοθούν συμβολισμοί, λήμματα και θεωρήματα, με τις αποδείξεις τους, καθώς και από ένα απλό παράδειγμα του τρόπου εκτέλεσής τους.

Στο τρίτο κεφάλαιο θα ασχοληθούμε με την υλοποίηση των δύο αλγορίθμων. Θα δοθούν λεπτομέρειες για την υλοποίησής τους, τη μορφή των αρχείων εισόδου και εξόδου τους καθώς επίσης θα παρουσιάσουμε τα βασικότερα τμήματα κώδικα και τις μεθόδους οι οποίες έχουν περιγραφεί στο δεύτερο κεφάλαιο. Τέλος με την οπτικοποίηση την οποία έχουμε υλοποιήσει, θα παρουσιάσουμε και σχηματικά τα αποτελέσματα των παραπάνω μεθόδων.

Το τέταρτο και τελευταίο κεφάλαιο θα είναι ο επίλογος, μαζί με κάποια συμπεράσματα στα οποία καταλήξαμε με το πέρας αυτής της διπλωματικής εργασίας.

2

Υπολογισμός Μικρότερου Περιγεγραμμένου Τριγώνου και Μεγίστου Εγγεγραμμένου Τριγώνου.

2.1 Υπολογισμός Μικρότερου Περιγεγραμμένου Τριγώνου

Ξεκινώντας αξίζει να αναφερθούμε σε κάποιους σημαντικούς ορισμούς, λέξεις κλειδιά, θεωρήματα και λήμματα τα οποία θα χρησιμοποιηθούν στη συνέχεια:

Βάση: όταν αναφερόμαστε στη βάση του τριγώνου θα αναφερόμαστε στην πλευρά C

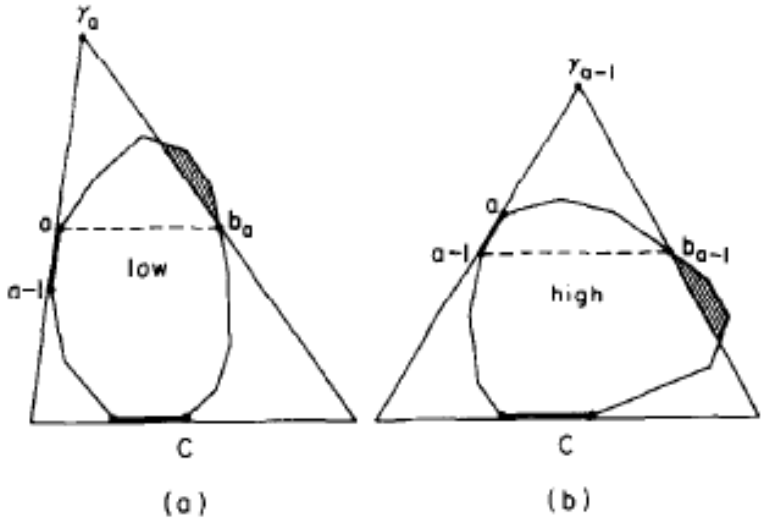
Αριστερή αλυσίδα: έστω h(p) η απόσταση του σημείου p από την πλευρά C, στην αριστερή αλυσίδα ανήκουν όλες οι κορυφές για τις οποίες ισχύει ότι $h(p) \leq h(p+1)$, όπου p+1 η επόμενη ωρολογιακά κορυφή. Συνεπώς όλες οι υπόλοιπες ανήκουν στη δεξιά αλυσίδα.

Χαμηλή: χαρακτηρίζεται μια πλευρά, έστω [a-1,a] του P όταν το $\gamma_a b_a$ τέμνει το P πάνω από το B (Σχήμα 2.1)

Υψηλή: χαρακτηρίζεται μια πλευρά, έστω [a-1,a] του P όταν το $\gamma_{a-1} b_{a-1}$ τέμνει το P κάτω από το B (Σχήμα 2.1)

Κρίσιμη: χαρακτηρίζεται μια πλευρά όταν δεν είναι ούτε χαμηλή ούτε υψηλή

P-anchored: χαρακτηρίζεται ένα τρίγωνο του οποίου η μια πλευρά εφάπτεται με το P καθώς και τα μέσα των άλλων δύο. Ένα P-anchored τρίγωνο μπορεί να μην είναι τοπικό ελάχιστο αλλά κάθε τοπικό ελάχιστο είναι P-anchored.



Σχήμα 2.1 *στο (α), η ακμή [a-1,a] είναι χαμηλή, ενώ στο (b) υψηλή* [1]

Τα επόμενα τρία θεωρήματα είναι μια πιο απλοποιημένη εκδοχή του γεωμετρικού χαρακτηρισμού των Klee και Laskowski, τα οποία δεχόμαστε χωρίς απόδειξη.

Θεώρημα 1 (Victor Klee): Αν το T είναι ένα τοπικό ελάχιστο μεταξύ των τριγώνων που περικλείουν το P, τότε τα μέσα κάθε ακμής του T εφάπτονται στο P.

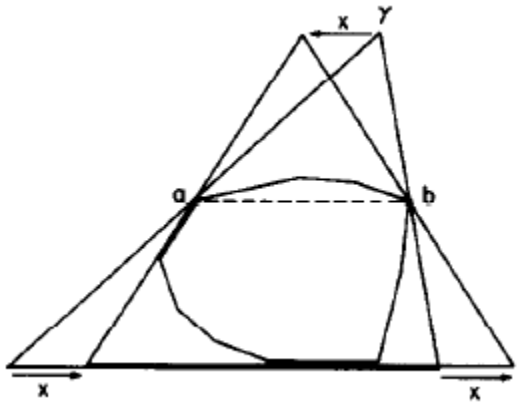
Θεώρημα 2 (Victor Klee και Michael C. Laskowski): Αν το T είναι ένα τοπικό ελάχιστο μεταξύ των τριγώνων που περικλείουν το P, τότε τουλάχιστον μια πλευρά του T είναι εφαπτομένη στο P. Κατά τη διάρκεια της διπλωματικής αυτή, η συγκεκριμένη ακμή θα είναι η C.

Θεώρημα 3 (Victor Klee και Michael C. Laskowski): Για να αυξήσουμε την απόσταση από την πλευρά C, τόσο η αριστερή όσο και η δεξιά αλυσίδα περιλαμβάνουν μια ακολουθία από χαμηλές ακμές, ακολουθούμενες από το πολύ δύο κρίσιμες, ακολουθούμενες από μια ακολουθία από υψηλές ακμές. Για κάθε εφαπτόμενη πλευρά C, υπάρχει ένα P-anchored τρίγωνο. Αν το ABC είναι P-anchored και έχει το C εφαπτόμενη πλευρά, τότε τα μέσα των πλευρών A και B βρίσκονται είτε σε κρίσιμη ακμή είτε σε μια κορυφή μεταξύ μιας χαμηλής και μιας υψηλής ακμής.

Λήμμα 1: Για κάθε P-anchored τρίγωνο T, υπάρχει ακόμα ένα με ίσο εμβαδό P-anchored T', στο ίδιο τμήμα του T, το οποίο έχει τουλάχιστον δύο ακμές εφαπτόμενες στο P, επομένως μπορούν να κατηγοριοποιηθούν στην ιδιά ομάδα.

- Απόδειξη: Έστω C η flush ακμή του T. Ας υποθέσουμε ότι ούτε η ακμή A ούτε η B είναι flush, και έστω ότι εφάπτονται με το P στις κορυφές a και b. Τώρα μετακινώντας την κορυφή γ του T αριστερόστροφα και παράλληλα στη C, όσο τα σημεία επαφής a και b παραμένουν σταθερά. Όπως

ευκολά μπορούμε να παρατηρήσουμε ότι το μήκος της βάσης του τρίγωνου παραμένει σταθερό και όσο το ύψος δεν αλλάζει, το εμβαδόν παραμένει σταθερό καθ' όλη τη μετακίνηση. Μετακινώντας το γ μέχρι η πλευρά A ή B να γίνει flush στο P. Αυτό το τρίγωνο είναι το T΄ (Σχήμα 2.2).



Σχήμα 2.2 *Λήμμα 1* [1]

Λήμμα 2: Έστω T = ABC είναι P-anchored τρίγωνο με την C εφαπτόμενη στο P, με a και b τα μέσα των πλευρών A και B και c την πιο ακραία ωρολογιακά ακμή της εφαπτομένης ακμής του P. Έστω C΄ εφαπτόμενο με το P στην αλυσίδα (c,a). Τότε αν T΄ = A΄B΄C΄ είναι P-anchored τρίγωνο με την C΄ εφαπτόμενη στο P με a΄ και b΄ τα μέσα των πλευρών A΄ και B΄ και c΄ την πιο ακραία ωρολογιακά ακμή της εφαπτομένης ακμής του P, ισχύει ότι b΄ ∈ (b, c΄) και a΄ ∈ (a, b΄) (Σχήμα 2.3)

- Απόδειξη: Από τη σύμβαση των ονομάτων, τα μέσα ωρολογιακά είναι τα a΄, b΄ και c΄. Έστω ότι τα ότι b΄ ∉ (b ,c΄) και/ή a΄ ∉ (a, b΄). Τότε, θα αποδείξουμε ότι το A΄B΄C΄ δεν μπορεί να είναι ένα P-anchored τρίγωνο θεωρώντας τις ακόλουθες τρεις περιπτώσεις (Σχήμα 2.3):

  Περίπτωση 1: b΄ ∈ [c΄ ,a]. Τότε, a΄ ∈ (c΄, b΄) έτσι ώστε τα a΄, b΄ και c΄ ανήκουν στην αριστερή αλυσίδα με βάση το C. Έτσι οι γωνίες μεταξύ των C΄ και A΄ και A΄ και B΄ είναι τουλάχιστον π/2 έτσι ώστε A΄B΄C΄ δε σχηματίζει τρίγωνο.

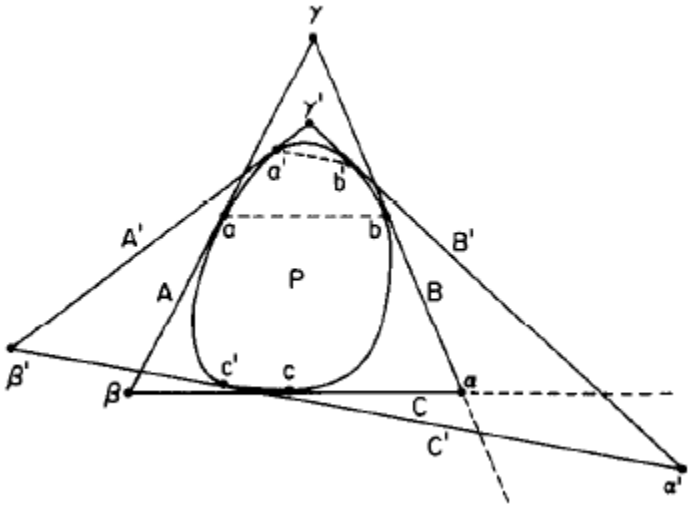
  Περίπτωση 2: b΄ ∈ [α ,b]. Τότε, θα αποδείξουμε το λήμμα θεωρώντας δύο υποπεριπτώσεις:

  Περίπτωση 2.1: Το σημείο τομής B΄ και Γ΄, α΄ , βρίσκεται στην ίδια πλευρά του B όπως το P. Εφόσον α' ∈ [c', b'], το τρίγωνο A'B'C' βρίσκεται εξ ολοκλήρου στη μία πλευρά του P, αλλά αυτό έρχεται σε αντίθεση με την υπόθεση μας.

  Περίπτωση 2.2: Το α΄ βρίσκεται στην πλευρά του B απέναντι από το P (Σχήμα 2.3). Εφόσον το A'B'C' είναι ένα P-anchored τρίγωνο, το τμήμα α΄b΄ είναι παράλληλο στο C΄. Συνεπώς, α΄∈(α, b΄),

υπονοώντας ότι το γ΄ βρίσκεται στο εσωτερικό του τριγώνου αγb. Τώρα, εφόσον το C' έχει περιστραφεί προς τα κάτω ως προς το C, είναι σαφές ότι το α΄ βρίσκεται κάτω από το C. Αλλά τότε έχουμε ότι και το γ΄ είναι κάτω από το γ και το α΄ είναι κάτω από το α, το οποίο σημαίνει ότι το μέσο b΄ του γ΄α΄ είναι κάτω από το μέσο b του γα. Αλλά αυτό έρχεται σε αντίθεση με την υπόθεση μας ότι το b΄∈(α, b).

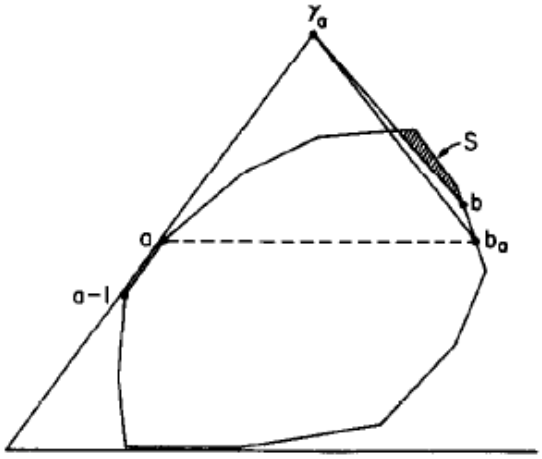
Περίπτωση 3: b΄∈ (b, c΄). Τότε το α΄∉ (α, b΄) (αλλιώς το λήμμα ικανοποιείται), ώστε το α΄ ∈ [c΄, α]. Τότε έχουμε και c΄ και α΄ στο [c, α], που είναι πανομοιότυπη με την Περίπτωση 2, με την Περίπτωση 2 c να έχει μετατραπεί ως β, και το β να έχει μετατραπεί ως α, κ.ο.κ.



Σχήμα 2.3 *Λήμμα 2* [1]

Λήμμα 3: αν h(b) > h(a) και γ_αb τέμνει το P από πάνω ή είναι εφαπτόμενο με το b, τότε η ακμή [a,a-1] είναι χαμηλή. (Σχήμα 2.4)

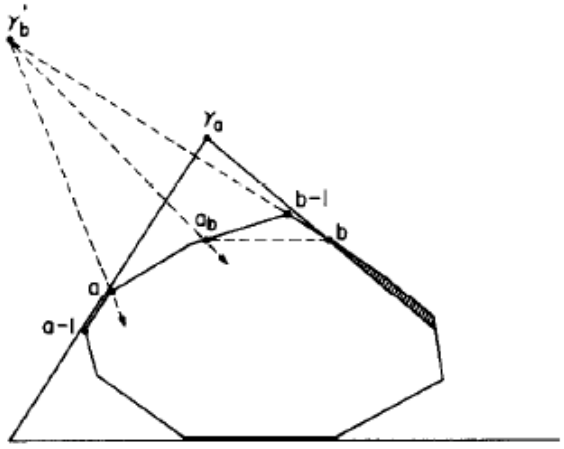
- Απόδειξη Έστω S το υποσύνολο του P που βρίσκεται πάνω από το γ_ab. (Σχήμα 2.4). Αφού h(b) > h(α), h(b) > h(b_α). Επομένως το S βρίσκεται επίσης πάνω από τα $y_\alpha b_\alpha$ και άρα η γραμμή αυτή κόβει P πάνω από το b_α. Με τον ορισμό του χαμηλού, η άκρη [a - 1, a] είναι χαμηλή.



Σχήμα 2.4 [1]

Λήμμα 4: αν h(b) > h(a) και γ_ab τέμνει το P από κάτω από το b, τότε η ακμή [b,b-1] είναι υψηλή. (Σχήμα 2.5)

- Απόδειξη Ας υποθέσουμε ότι το b - 1 βρίσκεται στη δεξιά αλυσίδα, διαφορετικά το [b - 1, b] είναι υψηλό. Θεωρήστε τη γραμμή B′ που καθορίζεται από το [b - 1, b] και ορίζουμε το γ'_b να είναι το σημείο στο B′ με $h(\gamma'_b) = 2h(b)$. Τότε το γ'_b πρέπει να βρίσκεται στα αριστερά της ευθείας που καθορίζεται από το [a - 1, a], αφού (1) h(b) > h(a), και έτσι $h(\gamma'_b) > h(\gamma_a)$ και (2) η κλίση του B′ είναι μικρότερη από εκείνη του γ'_bα(Σχήμα 2.5). Τώρα, είναι σαφές ότι η ευθεία γ'_bα τέμνει το P κάτω από το α. Αυτό συνεπάγεται, με ένα επιχείρημα παρόμοιο με αυτό που χρησιμοποιήθηκε στο προηγούμενο λήμμα, ότι το $\gamma'_b\alpha_b$ τέμνει το P κάτω από το α_b, όπου α_b είναι το σημείο στην αριστερή αλυσίδα με $h(\alpha_b) = h(b)$. Αλλά αυτό είναι ακριβώς ο ορισμός του τι σημαίνει η ακμή [b - 1, b] να είναι υψηλή.



Σχήμα 2.5 *Η ακμή [b,b-1] είναι υψηλή από το λήμμα 4* [1]

Σύνοψη**:**

P το πολύγωνο

T το τοπικό ελάχιστο

A, B, C οι πλευρές του τριγώνου, όπου C η βάση

a, b, c σημεία πάνω στις πλευρές A, B, C.

+ 1 επόμενη ωρολογιακά κορυφή – 1 προηγουμένη.

h(p) η απόσταση του σημείου p από την πλευρά C.

γ_p είναι το σημείο όπου ισχύει το $h(\gamma_p) = 2h(p)$.

2.1.1 Αλγόριθμος Υπολογισμού του Μικρότερου Τριγώνου Περιγεγραμμένου σε Δοθέν Κυρτό Πολύγωνο

a=1 b=2

Για c = 0,...,n

Μετακινούμε το b στη δεξιά αλυσίδα.

Όσο h(b + 1) ≥ h(b)

b = b + 1

Προχωράμε το a αν είναι χαμηλό και το b αν είναι υψηλό.

Όσο h(b) > h(a)

Αν γ_ab τέμνει το P κάτω από το b

b = b + 1

Αλλιώς

a = a + 1

Το [a - 1, a] μετά το τέλος της Όσο, είναι είτε κρίσιμο είτε υψηλό σημείο.

Ξεκινάμε την αναζήτηση για την εφαπτομένη B(τη δεύτερη πλευρά του τριγώνου).

Όσο γ_bb τέμνει το P κάτω από το b και h(b) ≥ h(a - 1)

b = b + 1

Αν γ_bb τέμνει το P πάνω από το b ή h(b) < h(a - 1)

Κάνουμε την πλευρά B «flush» με το τμήμα [b - 1, b].

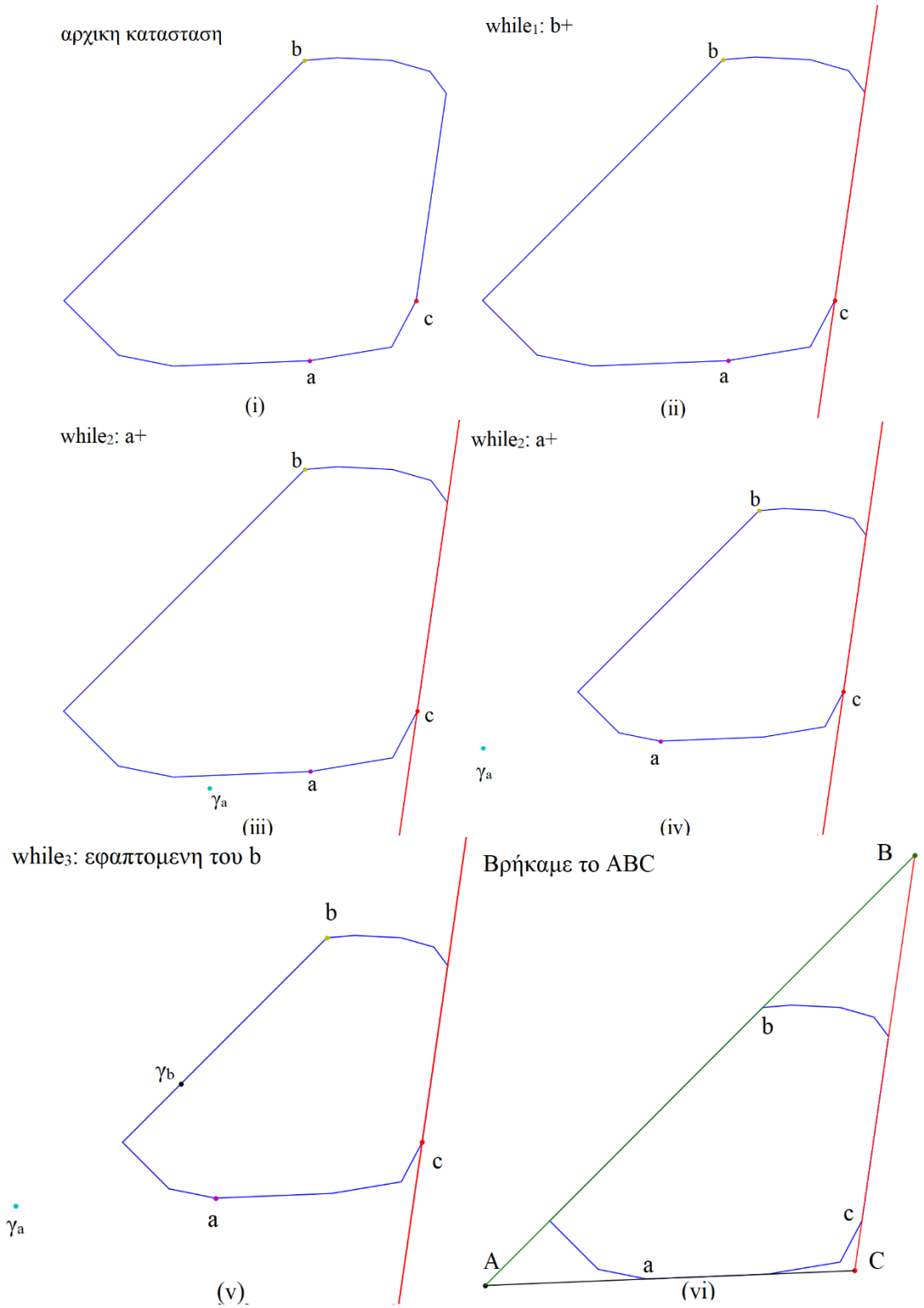
Αν το μέσω της πλευράς B < h(a - 1)

Τοποθετούμε την πλευρά A ώστε να έχει μεσώ το a – 1.

Αλλιώς

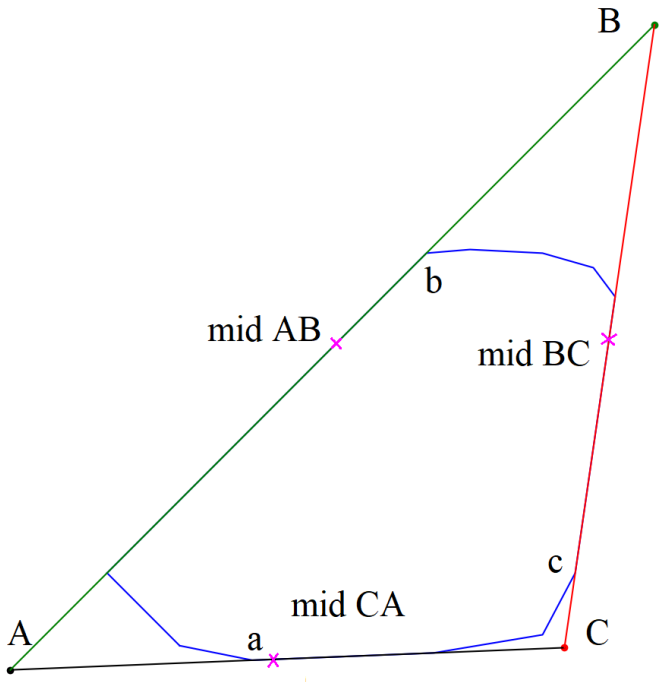
Η πλευρά B είναι η γ_bb.

Τέλος υπολογίζουμε το εμβαδόν του ABC, ενώνοντας τα σημεία τομής των τριών πλευρών.



Σχήμα 2.6 *Στιγμιότυπα από την εκτέλεση του αλγορίθμου*

Από το Σχήμα 2.6 μας δίνεται μια πρώτη εικόνα του τρόπου εκτέλεσης και τα βήματα του αλγορίθμου. Αναλυτικότερα, στο (i) είναι η αρχική κατάσταση, στο (ii) προχωράμε το b στη δεξιά αλυσίδα. Στη περίπτωσή μας το b τυγχάνει να είναι ήδη στην αλυσίδα και επομένως δε φαίνεται κάποια αλλαγή. Στο (iii) εκτελείται το δεύτερο while στο οποίο προχωράμε το a αν είναι χαμηλό και το b αν είναι υψηλό. Το αποτέλεσμά του γίνεται αντιληπτό στο (iv) στο οποίο βλέπουμε το a να έχει προχωρήσει προς την ωρολογιακή φορά μέχρι να μην είναι χαμηλό. Το επόμενο βήμα (v) είναι η εκτέλεση του τρίτου while όπου ξεκινάει η αναζήτηση για την εφαπτομένη B. Όσο το γ_bb τέμνει το P κάτω από το b προχωράμε το b. Το P δεν τέμνεται, επομένως δε βλέπουμε κάποια αλλαγή. Τέλος ενώνουμε τα σημεία που βρήκαμε, το τρίγωνο είναι P-anchored, και υπολογίζουμε το εμβαδόν του. Υπολογίζουμε τα μέσα κάθε πλευράς και αν είναι ενδιάμεσα στο τμήμα που εφάπτεται με το P, τότε το τρίγωνο είναι τοπικό ελάχιστο (Σχήμα 2.7)



Σχήμα 2.7 *Το ABC είναι τοπικό ελάχιστο αφού τα μέσα κάθε κορυφής είναι ενδιάμεσα στο τμήμα που εφάπτεται με το P*

Θεώρημα 4: Ο αλγόριθμος βρίσκει σωστά όλα τα τοπικά ελάχιστα περιγεγγραμμένα τρίγωνα σε χρόνο O(n)

- Απόδειξη Το θεώρημα 3 εγγυάται ότι υπάρχει P-anchored τρίγωνο για κάθε επανάληψη του βρόχου for. Εάν ένα P-anchored τρίγωνο βρεθεί σε μία επανάληψη, το Λήμμα 2 εγγυάται ότι χρειαζόμαστε να αναζητήσουμε μόνο δεξιόστροφα για το επόμενο P- anchored όταν το c προχωρά. Σίγουρα αυτό ισχύει στην πρώτη επανάληψη, όταν c = 0, a = 1 και b = 2. Το Λήμμα 1 εγγυάται ότι δεν θα χάσουμε

κανένα P-anchored τρίγωνο περιορίζοντας την αναζήτηση σε αυτά με τις C και A του τριγώνου να είναι flush. Ο πρώτος βρόχος while δεν είναι προβληματικός, καθώς το b πρέπει σαφώς να βρίσκεται στη δεξιά αλυσίδα και μετακινώντας το b σε κορυφή με μέγιστο ύψος δεν μπορεί να μετακινηθεί πέρα από μια κρίσιμη κορυφή (Σχήμα 2.8α).

Ο δεύτερος βρόχος while είναι ο πιο δύσκολος, αλλά το λήμμα 3 εγγυάται ότι το α προχωρά μόνο όταν το [a - 1, a] είναι γνωστό ότι είναι χαμηλό και ότι προχωρά το b μόνο όταν το [b - 1, b] είναι γνωστό ότι είναι υψηλό. Επομένως, αυτός ο βρόχος δεν μπορεί να μετακινήσει είτε το a είτε το b πέρα από μια κρίσιμη κορυφή. Έτσι διαπιστώνουμε ότι στο τέλος αυτού του δεύτερου βρόχου while, η άκρη [a - 1, a] δεν είναι χαμηλή.

Έστω το t ως το πιο δεξιόστροφο σημείο του P με την ιδιότητα ότι $\gamma_\alpha t$ υποστηρίζει το P στο t από πάνω, για ένα δεδομένο α, t το B είναι η εφαπτόμενη πλευρά. Εμείς ισχυριστείτε ότι, εάν το b προχωρήσει από τον δεύτερο βρόχο while, η σχέση $h(b) \geq h(t)$ θα διατηρείται από τον βρόχο. Όποτε ο δεύτερος βρόχος προχωρά b, το t είναι κάτω από το b αφού το $y_\alpha b$ τέμνει το P κάτω από το b (και το b βρίσκεται στη δεξιά αλυσίδα), και η προώθηση του b δεν κινείται ποτέ πέρα από την εφαπτομένη (Σχήμα 2.8c και 2.8d). Κάθε φορά που ο δεύτερος βρόχος προχωρά a, το t μετακινείται μόνο δεξιόστροφα (Σχήμα 2.8b και 2.8c). Αυτό καθορίζει ότι το t είναι πάντα ίσο ή δεξιόστροφα του b μετά την πρώτη προώθηση του b.

Διαπιστώνουμε ότι το [a - 1, a] δεν είναι χαμηλό στο τέλος του δεύτερου while μέσω αντίφαση, εξετάζοντας δύο περιπτώσεις.

Υποθέτουμε πρώτα ότι το [a - 1, a] είναι χαμηλό μετά το τέλος του βρόχου και ότι το ο βρόχος έχει προχωρήσει το b. Επειδή έχει τελειώσει, $h(b) \leq h(\alpha) = h(b_\alpha)$, επειδή [a - 1, a] είναι χαμηλό, $y_\alpha b_\alpha$ τέμνει το P πάνω από το b_α. Αυτό σημαίνει ότι το $y_\alpha b$ τέμνει το P πάνω από το b. Αλλά τότε το b είναι κάτω από το σημείο της εφαπτομένης του t, έρχεται σε αντίθεση με τη σχέση που δείξαμε παραπάνω: $h(b) \geq h(t)$.

Δεύτερον, υποθέτουμε ότι το [a - 1, a] είναι χαμηλό μετά τον βρόχο και ότι το b δεν προχώρησε από τον βρόχο. Τότε είτε το b μετακινήθηκε από το πρώτο while, είτε δεν μετακινήθηκε καθόλου κατά τη διάρκεια της επανάληψης του βρόχου for. Αν μετακινήθηκε από το πρώτο while, τότε είναι στο μέγιστο ύψος από το C, και είναι καθαρά αδύνατο το $h(a) \geq h(b)$ (η συνθήκη εξόδου για το δεύτερο while) και [a - 1, a] είναι ακόμα χαμηλό. Πρέπει λοιπόν το b να μην έχει μετακινηθεί ποτέ σε αυτήν την επανάληψη του βρόχου c. Σε αυτή την περίπτωση το λήμμα 2 μας λέει ότι η θέση ανάπαυσης για το b είναι δεξιόστροφα στην τρέχουσα θέση του. Αλλά αν $h(\alpha) \geq h(b)$ και το [a - 1, a] είναι χαμηλό, τότε το μέσο του A και επομένως το B πρέπει να βρίσκεται πάνω από το α και άρα πάνω από το b, το οποίο είναι αντίφαση.

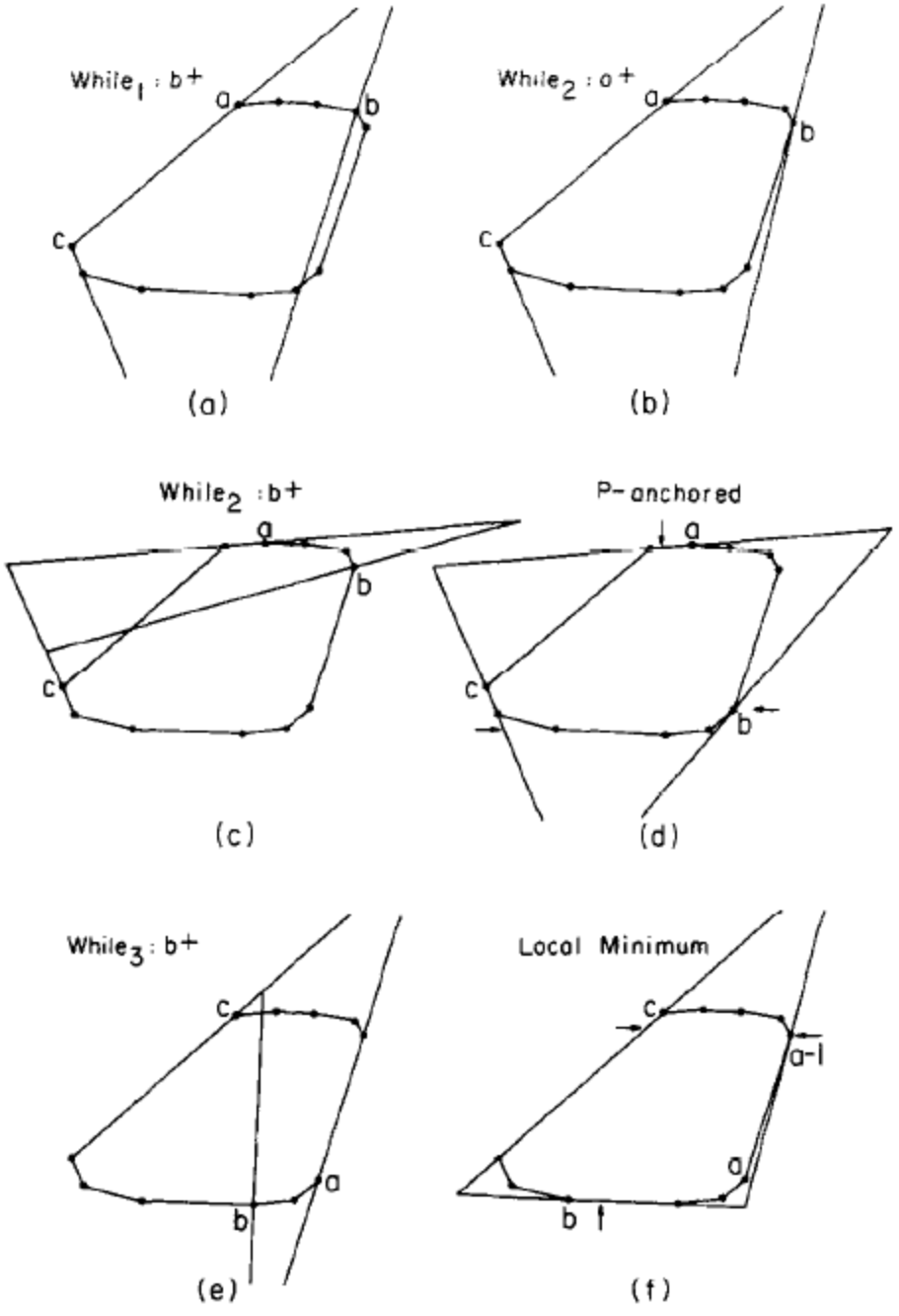
Εφόσον το [a - 1, a] δεν είναι χαμηλό, πρέπει να είναι κρίσιμο ή υψηλό. Έτσι δικαιολογείται να σταματήσει η προώθηση του α. Τώρα εξετάζουμε τη λειτουργία του τρίτου βρόχου while και ο υπόλοιπος κώδικας στο τέλος του βρόχου for και στις δύο περιπτώσεις κρίσιμες και υψηλές.

Εάν το [a - 1, a] είναι κρίσιμο, τότε εφόσον δεν είναι χαμηλό, το $y_\alpha b_\alpha$ δεν τέμνει από πάνω, και αφού δεν είναι ψηλό, το $y_{\alpha-1}b_{\alpha-1}$, δεν τέμνει από κάτω. Αυτό συνεπάγεται ότι υπάρχει κάποιο b_t, με $h(a - 1) \leq h(b_t) \leq h(\alpha)$ έτσι ώστε το $\gamma_{bt}b_t$ να εφάπτεται στο β_t. Επομένως το τρίτο while είτε θα βρει το b_t αν είναι κορυφή του P, είτε μια κορυφή πέρα από αυτήν δεξιόστροφα, αν το b_t βρίσκεται στο εσωτερικό μιας ακμής του P. Σε επόμενη περίπτωση, η if ακολουθάει το σετ B flush με το [b - 1, b]. Και στις δύο περιπτώσεις, βρίσκεται ένα σωστό P-anchored.

Τώρα εξετάζουμε την περίπτωση όταν το [a - 1, a] είναι υψηλό. Εδώ δεν είναι δυνατόν για το A να είναι flush, αλλά η άκρη A πρέπει να αγγίζει μόνο στο a - 1, που είναι η κορυφή που χωρίζει το χαμηλό από το υψηλό. Εφόσον το [a - 1, a] είναι υψηλό, το $\gamma_{\alpha-1}b_{\alpha-1}$ τέμνει το P κάτω από το b_α. Έτσι το πρώτο μέρος της τρίτης συνθήκης while θα είναι αληθής όταν $h(b) \geq h(a - 1)$. Επομένως, ο βρόχος εξέρχεται με $h(b) < h(a - 1)$, και τα ακόλουθα, αν το B γίνει flush με το [b - 1, b], με αποτέλεσμα να συρρέουν το ύψος του a - 1 και το A έτσι ώστε το a - 1 να είναι το μέσο του(Σχήμα 2.8f). Αυτό είναι πράγματι ένα P-anchored τρίγωνο.

Τέλος, υπολογίζουμε το εμβαδόν του P-anchored τριγώνου ABC. Αν και το τρίγωνο δεν είναι απαραίτητα ένα τοπικό ελάχιστο (π.χ. το μέσο του C μπορεί να μην βρίσκεται στο [c - 1, c] Σχήμα 2.8d) δε θα χαθούν ελάχιστα.

Είναι προφανές ότι ο αλγόριθμος είναι γραμμικός, αυξάνει μόνο τα a, b και c, και επομένως ολοκληρώνεται μέσα σε $3n = O(n)$ βήματα.


While1 : b+
a
b
c
(a)
While2 : a+
a
b
c
(b)
While2 : b+
a
b
c
(c)
P-anchored
a
b
c
(d)
While3 : b+
c
a
b
(e)
Local Minimum
c
a-1
a
b
(f)

Σχήμα 2.8 [1]

2.2 Υπολογισμός Μεγίστου Εγγεγραμμένου Τριγώνου

Παρακάτω θα δοθούν ορισμοί λέξεις κλειδιά, θεωρήματα και λήμματα τα οποία θα χρησιμοποιηθούν στη συνέχεια:

Legs: είναι οι υπόλοιπες δύο ακμές του τριγώνου, πέραν της βάσης.

P-supported: χαρακτηρίζεται ένα τρίγωνο του οποίου η βάση εφάπτεται με το P ή ένα από τα δύο legs εφάπτονται με το P και έχει ως μέσο μια κορυφή της ακμής με την οποία εφάπτεται.

Support Line: είναι η ευθεία που χωρίζει το επίπεδο στα δύο και περικλείει εξ ολοκλήρου το P σε ένα από τα δύο ημιεπίπεδα.

P-stable: ένα P-supported τρίγωνο είναι P-stable αν είτε:

- Η βάση του εφάπτεται με το P είτε,
- Ένα από τα legs εφάπτονται με κάποια ακμή του P και έχει ως μέσο μια κορυφή του.

Εσωτερικό Τρίγωνο: Έστω ένα P-supported τρίγωνο ABC, και έστω a b τα μέσα των legs BC και AC και c ένα σημείο στη βάση AB που εφάπτεται στο P. Το τρίγωνο είναι εσωτερικό του ABC. Η βάση του εσωτερικού τριγώνου είναι ab, οι υπόλοιπες ακμές είναι τα legs και η c είναι η κορυφή. (Σχήμα 2.9)

Λήμμα 2.1(Dobkin και Snyder):

- Αν t είναι ένα μέγιστο εγγεγραμμένο τρίγωνο σε δοθέν κυρτό πολύγωνο, τότε για κάθε κορυφή v του t, η ευθεία που περνάει από την v και παράλληλα με την απέναντι πλευρά του t, είναι support line για το P.
- Έστω ένα δοθέν κυρτό πολύγωνο P, υπάρχει ένα μέγιστο εγγεγραμμένο τρίγωνο στο P του οποίου οι κορυφές είναι ένα υποσύνολο των κορυφών του P.

Λήμμα 2.2: Έστω ένα δοθέν κυρτό πολύγωνο P και μια supporting line L, και έστω S το σετ με τα P-supported τρίγωνα των οποίων οι βάσεις εφάπτονται με την L:

- Το σετ S δεν είναι άδειο και όλα τα στοιχεία στο S έχουν το ίδιο εμβαδόν καθώς και τα legs τους μοιράζονται τα ίδια μέσα.
- Εάν το σετ S περιέχει περισσότερα από ένα στοιχεία τότε τα μέσα των a και b είναι κορυφές του P και το σετ των κορυφών του S σχηματίζει ένα ευθύγραμμο τμήμα παράλληλο της L και με διπλάσιο ύψος από την L σε σχέση με το ευθύγραμμο τμήμα ab. Τα πιο ακραία ωρολογιακά σημεία του τμήματος αυτού μπορούν να βρεθούν σε σταθερό χρόνο O(1).

Λήμμα 2.3: Έστω ένα δοθέν κυρτό πολύγωνο P, και έστω T ένα P-supported τρίγωνο και έστω το t είναι εσωτερικό του T, τότε:

- Το t προσδιορίζεται μοναδικά από τον προσανατολισμό της βάσης του T

- Η βάση του T και του t είναι παράλληλες
- Το εμβαδόν του T είναι τέσσερις φορές μεγαλύτερο από αυτό του t.

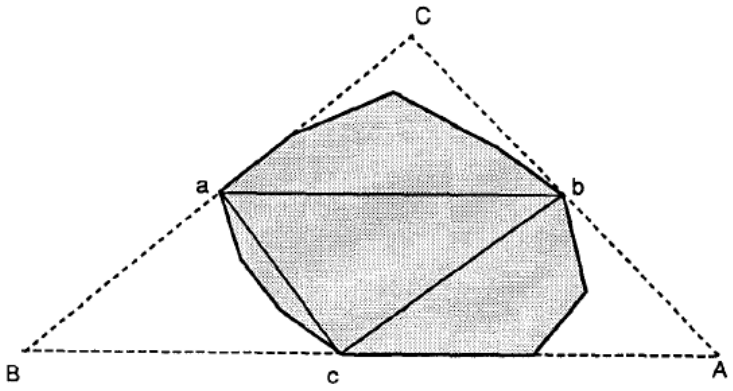
Λήμμα 2.4:

- υπάρχει ένα ελάχιστο περιγεγραμμένο τρίγωνο το οποίο είναι P-stable
- υπάρχει ένα μέγιστο εγγεγραμμένο τρίγωνο το οποίο είναι εσωτερικό ενός P-stable τριγώνου

Απόδειξη Η πρώτη παρατήρηση είναι μια άμεση συνέπεια του γεγονότος ότι όλα τα P-anchored τρίγωνα είναι P-stable. Για να αποδείξουμε τη δεύτερη παρατήρηση, έστω abc ένα μέγιστο κλειστό τρίγωνο έτσι ώστε τα a, b και c να είναι κορυφές του P. Σχηματίζουμε ένα τρίγωνο T = ABC τραβώντας μια γραμμή σε κάθε κορυφή του abc που είναι παράλληλη στην πλευρά απέναντι από την κορυφή. Από την πρώτη παρατήρηση στο Λήμμα 2.l, οι πλευρές του τριγώνου T είναι supporting line του P, επομένως το T περικλείει το P. Είναι εύκολο να προκύψει από τη στοιχειώδη γεωμετρία ότι οι πλευρές του T έχουν τα μέσα τους στα a, b και c. (Για παράδειγμα, το c είναι το μέσο του AB εάν παρατηρήσουμε ότι τα abAc και abcB είναι παραλληλόγραμμα που μοιράζονται την κοινή ακμή ab). Τώρα φέρνουμε το abc σε κανονική μορφή (αν χρειάζεται) μετακινώντας το c παράλληλα με το ab. Αυτό δεν αλλάζει το εμβαδόν του abc. Ομοίως μετατρέπουμε το T σε κανονική μορφή (αν χρειάζεται) ολισθαίνοντας την κορυφή C παράλληλα στην ακμή AB. Ας υποθέσουμε χωρίς βλάβη της γενικότητας ότι η πλευρά BC γίνεται flush με το P. Δεδομένου ότι το α είναι μια κορυφή του P και αφού το BC είναι flush με το P, το T είναι ένα κανονικό P-stable τρίγωνο και το abc είναι το εσωτερικό του τρίγωνο.

Έτσι, για τον προσδιορισμό του ελάχιστου περιγεγραμμένου τριγώνου και του μέγιστου εγγεγραμμένου τριγώνου αρκεί να υπολογίσουμε το σύνολο των κανονικών P-stable τριγώνων (που θα δείξουμε αργότερα ότι είναι ένα πεπερασμένο σύνολο) και να επιλέξουμε το εξωτερικό τρίγωνο του ελάχιστου εμβαδού και το αντίστοιχο εσωτερικό τρίγωνο μέγιστης επιφάνειας. Για να προχωρήσουμε, πρέπει πρώτα να εισαγάγουμε την ακόλουθη ιδιότητα, η οποία είναι βασική σε όλους τους αλγόριθμους rotating calipers[3]. Ο προσανατολισμός της βάσης προχωράει δεξιόστροφα, στις κορυφές του αντίστοιχου εσωτερικού τριγώνου όπου κινούνται δεξιόστροφα στο P. Αυτό το λήμμα δε δηλώνεται ακριβώς με αυτή τη μορφή από τους O'Rourke, et al. [1] αλλά προκύπτει άμεσα από την απόδειξη του Λήμματος 2 τους, μαζί με μια απλή παρατήρηση, ότι καθώς ο προσανατολισμός της βάσης στρέφεται δεξιόστροφα, το σημείο επαφής με τη βάση, c, κινείται επίσης δεξιόστροφα γύρω από το P.

Μπορούμε να συσχετίσουμε οποιοδήποτε κατευθυνόμενο ευθύγραμμο τμήμα AB με μια γωνία προσανατολισμού λαμβάνοντας υπόψη τη γωνία του διανύσματος B - A



Σχήμα 2.9 *Το εσωτερικό τρίγωνο abc* [7].

2.2.1 Αλγόριθμος Υπολογισμού του Μέγιστου Τριγώνου Εγγεγραμμένου σε Δοθέν Κυρτό Πολύγωνο

Σε συνέχεια του προηγούμενου αλγορίθμου (2.1.1) και στο εσωτερικό της Για, υπολογίζουμε το μέγιστο εσωτερικό τρίγωνο. Για τον υπολογισμό αυτόν διακρίνουμε τρεις περιπτώσεις.

Πρώτη περίπτωση

Η πρώτη περίπτωση είναι η απλούστερη. Ενώνουμε τα σημεία a, b και c στα οποία είχε σταματήσει ο πρώτος αλγόριθμος (2.1.1), με μια μόνο παρατήρηση.

Αν η παράλληλη με τα a, b που περνάει από το τρίτο σημείο τέμνει το P

το εμβαδόν μπορεί να αυξηθεί περνώντας στην επόμενη ωρολογιακά κορυφή του τρίτου σημείου.

Δεύτερη περίπτωση

Η δεύτερη περίπτωση είναι το περιγεγραμμένο τρίγωνο να έχει 2 flush legs (Σχήμα 2.10).

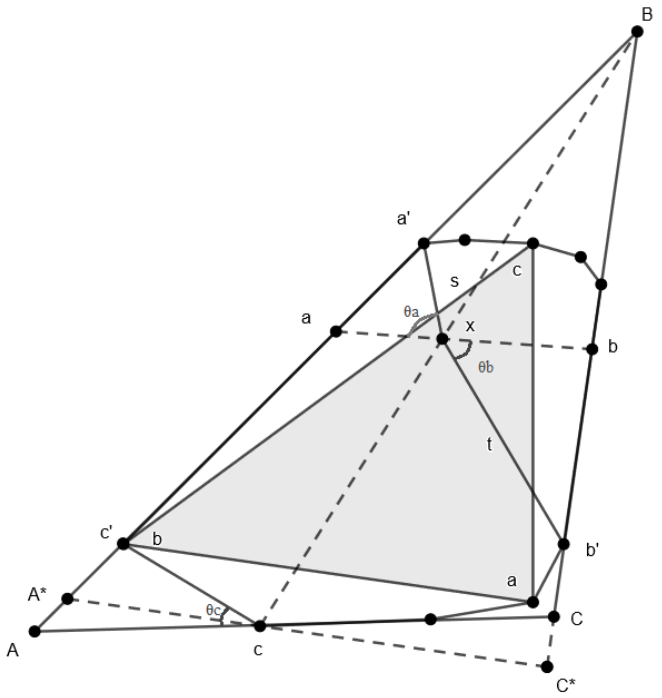
Έστω x το μέσο της κορυφής B με την κορυφή c και a, b τα μέσα των AB και AC.

Τα επόμενα τρίγωνα καθορίζονται από τις γωνίες που ορίζουν τα θ_a=axa’ θ_b=Acy θ_c=Acc’(όπου a’,b’,c’ είναι οι επόμενες ωρολογιακά κορυφές).

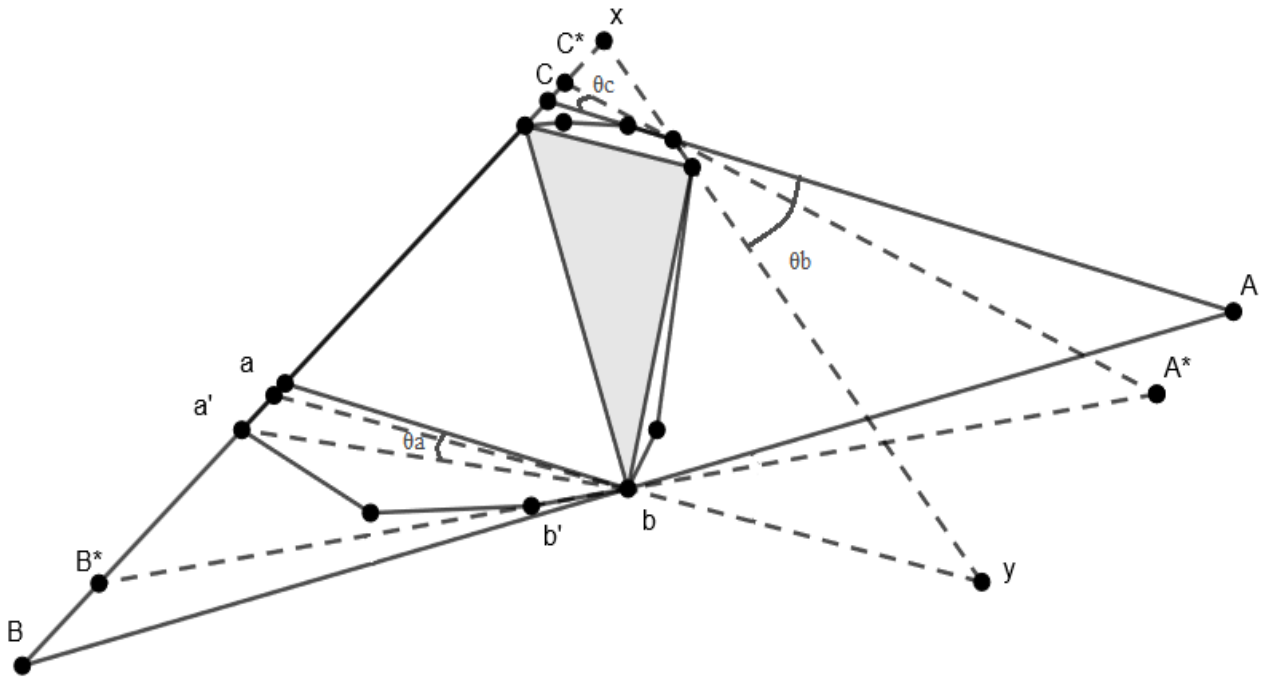
Τρίτη περίπτωση

Τρίτη και τελευταία περίπτωση είναι το περιγεγραμμένο τρίγωνο να έχει 1 flush leg (Σχήμα 2.11).

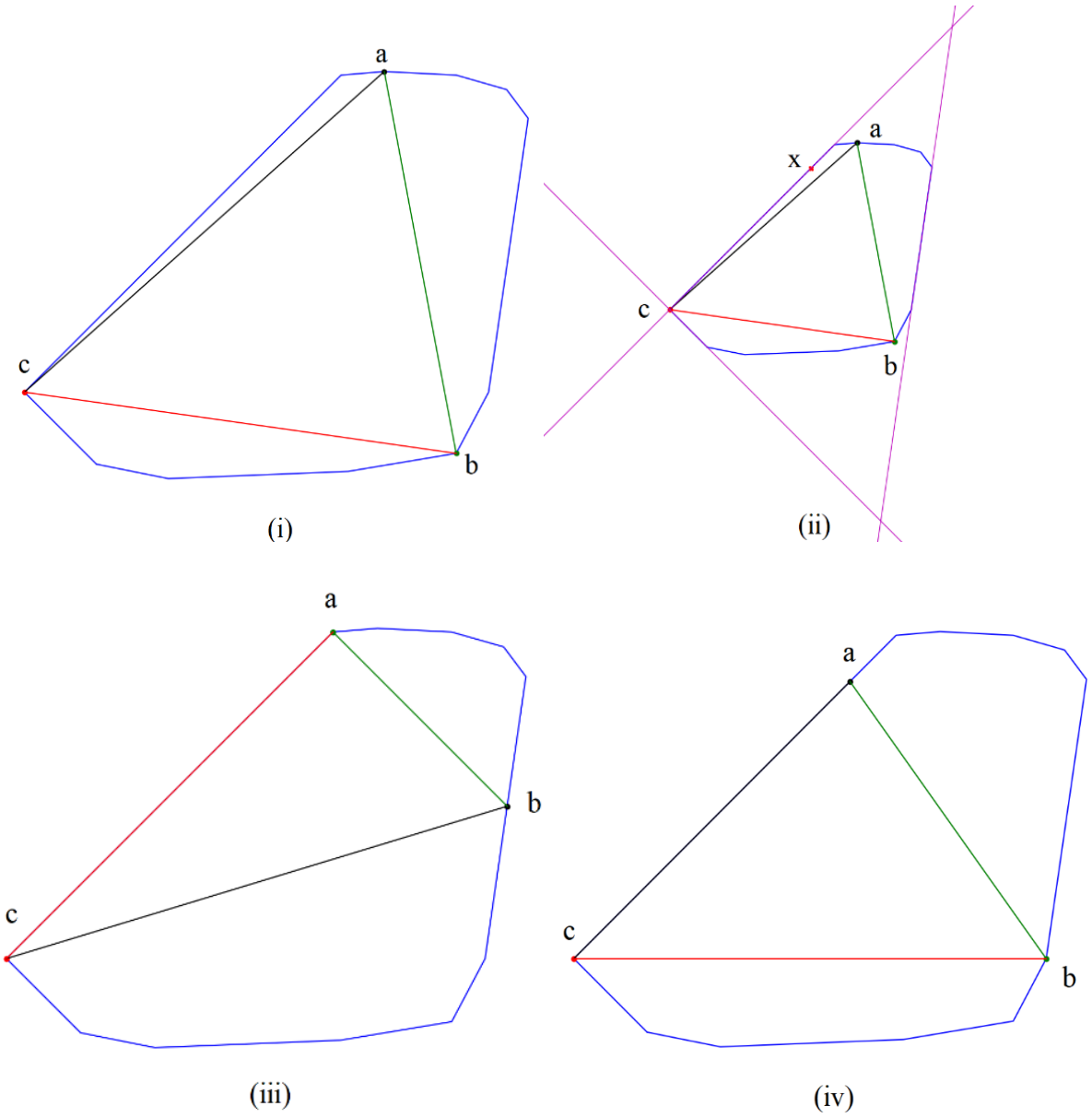
Τα επόμενα τρίγωνα καθορίζονται από τις γωνίες που ορίζουν τα θ_a=aba’ θ_b=Acy’ θ_c=C*cC(όπου a’,b’,c’ είναι οι επόμενες ωρολογιακά κορυφές).



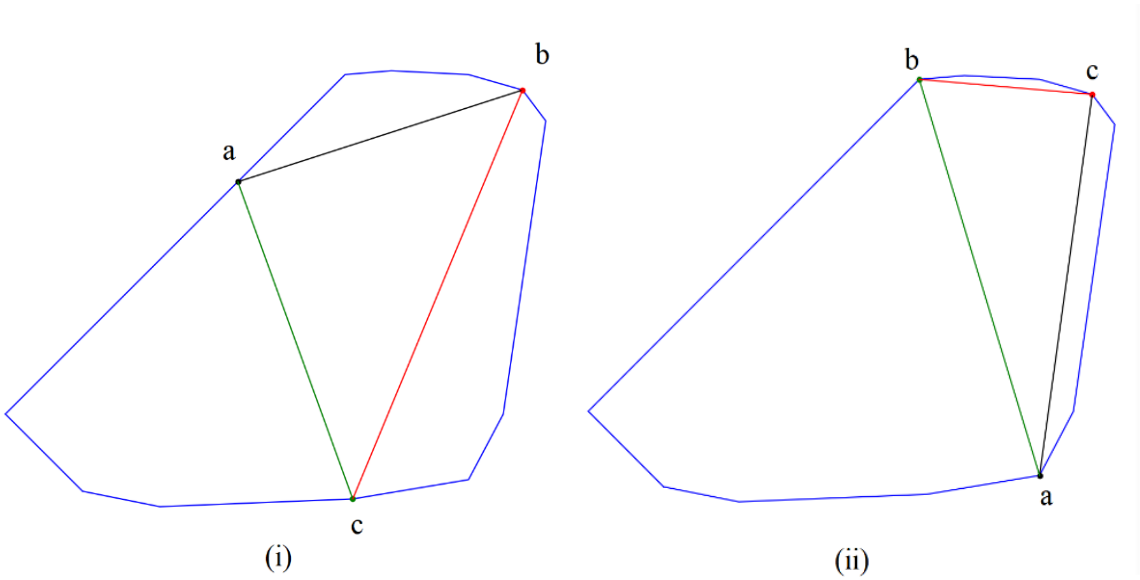
Σχήμα 2.10 *Δεύτερη περίπτωση 2 flush legs*



Σχήμα 2.11 *Τρίτη περίπτωση 1 flush leg*



Σχήμα 2.12 *Στιγμιότυπα από τον αλγόριθμο για την πρώτη και δεύτερη περίπτωση*



Σχήμα 2.13 *Στιγμιότυπα από τον αλγόριθμο για την τρίτη περίπτωση*

Από τα Σχήματα 2.12 και 2.13 βλέπουμε τον τρόπο εκτέλεσης και τα βήματα του αλγορίθμου για τον υπολογισμό του μέγιστου εγγεγραμμένου τριγώνου. Αναλυτικότερα, για το Σχήμα 2.12, (i) είναι η αρχική περίπτωση, οπού παίρνουμε τις πιο ακραία ωρολογιακά κορυφές των ακμών που βρισκόμαστε. Στο (ii) υπολογίζουμε το x και για τις γωνίες που προκύπτουν βρίσκουμε τα αντίστοιχα τρίγωνα. Συγκεκριμένα το (ii) είναι για το θ_c το οποίο εφόσον η πλευρά C εφάπτεται στο P είναι μηδενική επομένως δε βλέπουμε κάποια αλλαγή στο σχήμα. Το (iii) είναι για το θ_b και το (iv) για το θ_a. Για το Σχήμα 2.13, το (i) είναι η αρχική περίπτωση, οπού παίρνουμε τις πιο ακραία ωρολογιακά κορυφές των ακμών που βρισκόμαστε στο (ii) είναι για το θ_c. Για όλες τις πιθανές γωνίες που δημιουργούνται τα τρίγωνα που προκύπτουν ελέγχονται αν είναι έγκυρα, δηλαδή να βρίσκονται εξ ολοκλήρου εσωτερικά του P, για αυτό το λόγο στο Σχήμα 2.13 προκύπτουν λιγότερα τρίγωνα από ότι στο Σχήμα 2.12.

3

Η Υλοποίηση

Οι αλγόριθμοι υλοποιήθηκαν χρησιμοποιώντας ως γλώσσα προγραμματισμού την Python η οποία είναι μια υψηλού επιπέδου γλώσσα. Το χαρακτηριστικό που την ξεχωρίζει από τις υπόλοιπες, είναι η αναγνωσιμότητα που προσφέρει με το συντακτικό της και η ευκολία στη χρήση της. Επίσης διακρίνεται για την ταχύτητα εκμάθησης καθώς και για το πλήθος των βιβλιοθηκών της, οι οποίες διευκολύνουν αρκετά συνηθισμένες διεργασίες και συνήθως αναφέρονται ως το δυνατότερο σημείο της γλώσσας. Μια από αυτές είναι η Matplotlib, η οποία είναι απαραίτητη προϋπόθεση για την λειτουργεία του κώδικά μας. Η Matplotlib είναι μια open-source βιβλιοθήκη η οποία υποστηρίζει διαφόρων ειδών γραφήματα. Μπορεί να δημιουργήσει γραφήματα, ιστογράμματα, ραβδογράμματα και άλλους τύπους γραφημάτων με λιγοστές γραμμές κώδικα. Στον κώδικά μας είναι υπεύθυνη για την οπτικοποίησή του. Τέλος από την βιβλιοθήκη Math εκμεταλλευόμαστε τα sqrt, cos, sin, asin, degrees και pi τα οποία αναλαμβάνουν τα μαθηματικά.

3.1 Είσοδος – Έξοδος

Η είσοδος του κώδικά μας είναι ένα αρχείο κειμένου τύπου txt (polygon.txt) το οποίο ανήκει στον ίδιο φάκελο με τον κώδικα. Στο εσωτερικό του βρίσκουμε τα κυρτά πολύγωνα για τα οποία μας ενδιαφέρει να υπολογίσουμε τα ελάχιστα και μέγιστα περιγεγγραμμένα και εγγεγραμμένα τρίγωνα αντίστοιχα. Τα πολύγωνα πρέπει να δοθούν με την μορφή λίστας η οποία αποτελείται από τις συντεταγμένες των κορυφών του. Η λίστα θα πρέπει να είναι της μορφής [[…],[…],…,[…]]. Εάν θέλουμε να ελέγξουμε περισσότερα από ένα πολύγωνα, αυτά μπορούν να δοθούν με τον ίδιο τρόπο, αρκεί να βρίσκονται σε μια νέα γραμμή. Ουσιαστικά τα πολύγωνα χωρίζονται μεταξύ τους με τον τελεστή αλλαγής γραμμής “\n”

Η έξοδος του κώδικα είναι επίσης ένα αρχείο κειμένου τύπου txt (results.txt) το οποίο ανήκει στον ίδιο φάκελο με τον κώδικα. Το αρχείο αυτό περιέχει το πολύγωνο, τα ελάχιστα και μέγιστα τρίγωνα, τα οποία αποτελούνται από τα εμβαδά τους και τις συντεταγμένες. Εάν το αρχείο δεν υπάρχει, δημιουργείται και

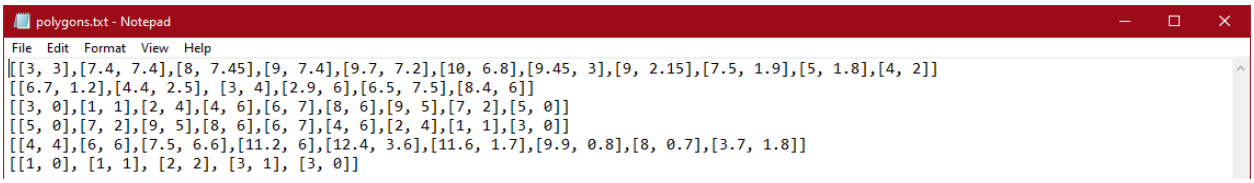
προσθέτουμε κάθε φορά το αποτέλεσμα του αλγορίθμου σε αυτό. Τέλος στο τερματικό εμφανίζονται κατάλληλα μηνύματα, έτσι ώστε να γνωρίζουμε σε πιο σημείο του κώδικα βρισκόμαστε.

3.2 Υλοποίηση Main

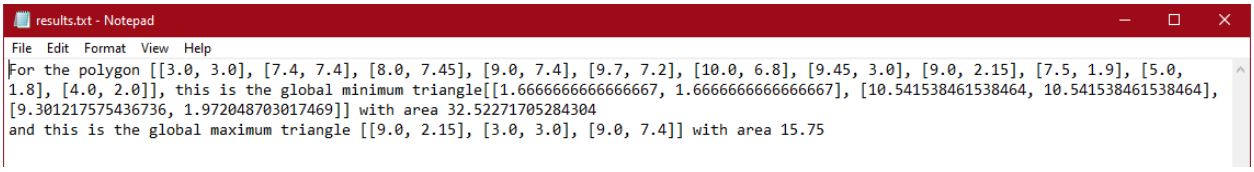
3.2.1 main()

```
def main():
    # ena arxeio tis morfis [[...],[...],...,[...]] me tis sintetagmentes twn
    # polygwnwn pou theloyme ma vroyme ta megista kai elaxista trigwna
    with open("polygon.txt", "r") as f:
        contents = f.read()
        # katharizoyme ta contents apo ta "[" , "]"
        contents = contents.replace("[", " ")
        contents = contents.replace("]", " ")
        # xwrizoume se lines ta contents, kathe line exei apo ena polygwno
        lines = contents.split("\n")
        polygons = []
        for line in lines:
            # se periptosi pou exei ksexastei enas kenos xaraktiras
            # sto telos i se kapoio simeio toy txt arxeioy
            if not line:
                continue
            coordinates = line.split(",")
            polygon = []
            # ana dyo prosthetoyme ta coordinates toy line se mia lista
            for index, _ in enumerate(coordinates):
                if index % 2 != 0:
                    continue
                polygon.append(
                    [float(coordinates[index]), float(coordinates[index + 1])]
                )
            polygons.append(polygon)
    # gia kathe ena polygwno ypologizoyme min kai max triangles
    for polygon in polygons:
        # constrained gia na min yparxoun polla kena sto figure
        print(polygon)
        plt.figure(figsize=(8, 8), layout="constrained", dpi=100)
        n = len(polygon)
        a = 1
        b = 2
        # se periptwsi pou den dwthoun se wrologiaki fora oi koryfes
        # allazw 'fora' etsi wste na einai
        polygon = clockwise_order(polygon, n)
        global_minimum, global_maximum = minimal_enclosing_triangles(a, b, n, polygon)
       # ta apotelesmata ta pername se ena arxeio results.txt
        with open("results.txt", "w") as f:
            f.write("For the polygon {polygon}, this is the global minimum
triangle{global_minimum[1:]} with area {global_minimum[0]} \
                \nand this is the global maximum triangle {global_maximum[1:]} with
area {global_maximum[0]} \n\n")
    plt.show()
if __name__ == "__main__":
    main()
```

Αρχικά έχουμε προσθέσει την προϋπόθεση να τρέχει ο κώδικας μόνο εάν έχει κληθεί από το ίδιο το αρχείο, με την προσθήκη της συνθήκης if που βρίσκεται στο κάτω μέρος. Η main() διαβάζει από το αρχείο polygons.txt, το οποίο πρέπει να βρίσκεται στον ίδιο φάκελο με τον κώδικα, αλλιώς πρέπει να δοθεί το κατάλληλο μονοπάτι (path). Στο εσωτερικό του βρίσκονται τα κυρτά πολύγωνα που μας ενδιαφέρει να υπολογίσουμε τα ελάχιστα και μέγιστα περιγεγγραμμένα και εγγεγραμμένα τρίγωνα αντίστοιχα. Η δομή του αρχείου φαίνεται στο Σχήμα 3.1. Αφού επεξεργαστούμε τα δεδομένα του και δημιουργήσουμε μια λίστα από πολύγωνα, εάν υπάρχουν περισσότερα από ένα, η for ξεκινάει τον υπολογισμό των ελάχιστων και μέγιστων τριγώνων, καθώς επίσης δημιουργεί και τα γραφήματα για την οπτικοποίηση. Μια ακόμα εργασία που αναλαμβάνει είναι η μετατροπή των κορυφών του πολυγώνου σε ωρολογιακή φορά, εάν αυτή δεν έχει δοθεί έτσι. Τέλος δημιουργεί ένα αρχείο results.txt (Σχήμα 3.2), εάν δεν υπάρχει, στο οποίο αποθηκεύει τα εμβαδά των τριγώνων και τις συντεταγμένες των κορυφών τους.



Σχήμα 3.1 *Δομή αρχείου polygons.txt*



Σχήμα 3.2 *Δομή αρχείου results.txt*

3.3 Υλοποίηση Αλγόριθμου 1

3.3.1 minimal_enclosing_triangles()

```
def minimal_enclosing_triangles(a, b, n, polygon):
    alpha = 0
    beta = 0
    gamma = 0
    global_minimum = None
    global_maximum = None

    for c in range(n):
        clear_plot(a, b, c, polygon)
```

```
        animation_step("Advance b to right chain")

        # Advance b to right chain
        b = advance_b_to_right_chain(b, c, polygon, n)

        draw_figure(a, b, c, polygon)
        c_line_x, c_line_y = line(c, polygon)

        animation_step(" Draw line C")
        plt.axline(c_line_x, c_line_y, linewidth=2, color="r")

        intersection_BC = intersecting_lines(
            polygon[b],
            polygon[b - 1],
            polygon[c],
            polygon[c - 1],
        )

        # elegxoume to intersection point tis pleyras B kai C
        # an den yparxei (intersection_BC == 0) de synexizw ton ypologizmo to trigwnoy
        # afoy oi dyo pleures einai paralliles
        if not intersection_BC:
            b_line_x, b_line_y = line(b, polygon)
            plt.axline(b_line_x, b_line_y, linewidth=2, color="r")
            animation_step("There is no local triangle.\n Sides B and C are parallel")
            continue

        # Move a if low, and b if high
        # gamma a
        ya = y(polygon[a], a, c, polygon)

        animation_step("Calculate ya")
        draw_figure(a, b, c, polygon, ya, ya_options=["c", "ya"])

        a, b, ya = move_a_if_low_and_b_if_high(a, b, c, ya, polygon, n)
        animation_step("Move a if low and b if high")

        draw_figure(a, b, c, polygon, ya, ya_options=["c", "ya"])

        # Search for the B tangency
        # gamma b
        yb = y(polygon[b], b, c, polygon)

        animation_step("Calculate yb")

        draw_figure(a, b, c, polygon, ya, yb, ya_options=["c", "ya"],yb_options=["k",
"yb"],
        )

        alpha, mid_alpha, b, beta, yb, flush_b = search_for_B_tangency(
            a, b, c, yb, alpha, beta, polygon, n
        )

        animation_step("Search for the B tangency")

        draw_figure(a, b, c, polygon, ya, yb, ya_options=["c", "ya"],
            yb_options=["k", "yb"],
        )

        alpha, beta, gamma = triangle_points(b, c, alpha, beta, mid_alpha, polygon)

        print(abs(triangle_area(alpha, beta, gamma)))
```

```
        area = draw_triangle(alpha, beta, gamma, polygon)
        if local_minimum(alpha, beta, gamma, a, b, c, polygon):
            add_suptitle("Triangle is local minimum")
            if not global_minimum or area < global_minimum[0]:
                global_minimum = [area, alpha, beta, gamma]
        else:
            if not global_minimum or area < global_minimum[0]:
                global_minimum = [area, alpha, beta, gamma]
            add_suptitle("Triangle is not local minimum")

        # -------------------------maximal enclosed triangle-------------------------
        global_maximum = maximal_enclosed_triangle(
            a, b, c, alpha, beta, gamma, mid_alpha, flush_b, n, global_maximum,
polygon
        )
        # -------------------------maximal enclosed triangle-------------------------

    if global_minimum:
        alpha, beta, gamma = global_minimum[1:]
        draw_triangle(alpha, beta, gamma, polygon)
        add_suptitle(f"This is the global minimum for this polygon")

    if global_maximum:
        inner_a, inner_b, inner_c = global_maximum[1:]
        plt.figure(figsize=(8, 8), layout="constrained", dpi=100)

        draw_triangle(inner_a, inner_b, inner_c, polygon)
        add_suptitle(f"This is the global maximum inner triangle for this polygon")

    return global_minimum, global_maximum
```

Η minimal_enclosing_triangles() ήταν αρχικά υπεύθυνη για το υπολογισμό του ελάχιστου περιεγραμμένου τριγώνου, όμως λόγω των πολλών κοινών σημείων μεταξύ των δύο αλγορίθμων το αποτέλεσμά της και πολλοί από τους υπολογισμούς που αναλάμβανε ήταν χρήσιμοι και στον υπολογισμό του μέγιστου εγγεγραμμένου, έτσι στο τέλος καλεί με αυτά ως ορίσματα την maximal_enclosed_triangle(). Επίσης ελέγχει αν ένα τρίγωνο είναι τοπικό ελάχιστο και αν ναι αποθηκεύει αυτό με το ελάχιστο εμβαδόν με σκοπό τον υπολογισμό του ολικού ελάχιστου περιεγραμμένου. Ανάλογη δουλειά κάνει και στην περίπτωση αναζήτηση του μέγιστου εγγεγραμμένου. Ένας επιπλέον έλεγχος γίνεται στην περίπτωση όπου οι δύο πλευρές που επεξεργαζόμαστε είναι παράλληλες. Συγκεκριμένα, εάν δεν υπάρχει σημείο τομής μεταξύ τους σταματάμε τους ελέγχους, εμφανίζοντας το κατάλληλο μήνυμα, και προχωράμε στην επόμενη επανάληψη της for. Τέλος επιστρέφει το ολικό ελάχιστο και μέγιστο τρίγωνο με την μορφή λίστας εμβαδού, συντεταγμένων.

Αναλυτικότερα στο εσωτερικό της βρίσκονται οι εξής συναρτήσεις:

3.3.2 clear_plot()

Καθαρίζει το plot μετά από κάθε επανάληψη της for έτσι ώστε το γράφημα να είναι καθαρό από προηγουμένους υπολογισμούς.

3.3.3 animation_step()

Για να φαίνεται βήμα βήμα η διαδικασία. Προστέθηκε έτσι ώστε για να προχωρήσει ο αλγόριθμος θα πρέπει ο χρήστης να πατήσει κάποιο κουμπί εισόδου, όπως το enter, mouse click κ.τ.λ. Επίσης εμφανίζει και ως τίτλο το βήμα στο οποίο βρισκόμαστε.

3.3.4 advance_b_to_right_chain()

Εάν το b δεν είναι εντός της δεξιάς αλυσίδας προχωράμε το b στην επόμενη ωρολογιακά κορυφή και επιστρέφει το νέο b.

3.3.5 draw_figure()

Εμφανίζει τις κορυφές που βρισκόμαστε με το αντίστοιχο χρώμα και στις συντεταγμένες τους καθώς και την supporting line η οποία όπως έχουμε αναφέρει παραπάνω είναι η C.

3.3.6 intersecting_lines()

Υπολογίζει και επιστρέφει το σημείο τομής δύο ευθειών, εάν υπάρχει αλλιώς επιστρέφει μηδέν.

3.3.7 y()

Υπολογίζει και επιστρέφει το γ_p του σημείου p, το οποίο έχει το διπλάσιο ύψος από το σημείο p. Εάν δεν υπάρχει το σημείο επιστρέφει την κορυφή p.

3.3.8 move_a_if_low_and_b_if_high()

Εάν το a βρίσκεται σε χαμηλή ακμή ή/και το b σε υψηλή προχωράμε ωρολογιακά τα a ή/και b μια θέση. Τέλος επιστρέφει τα a, b και γ_α.

3.3.9 search_for_B_tangency()

Υπολογίζουμε την εφαπτομένη B. Εάν το ευθύγραμμο τμήμα γ_bb τέμνει το P κάτω από το b και η απόσταση του b από την C είναι μεγαλύτερη από αυτή της a-1, προχωράμε το b στην επόμενη ωρολογιακά κορυφή. Στη συνέχεια υπάρχουν τρεις δείκτες οι οποίοι μας επιτρέπουν σε παρακάτω βήματα να μπορούμε να ελέγξουμε σε πια περίπτωση βρισκόμαστε, δηλαδή εάν τέμνει το P πάνω από το b, εάν θέλουμε η κορυφή a-1 να είναι το μέσο της A και αν η πλευρά B πρέπει να εφάπτεται με το P. Σε περίπτωση που το γ_bb τέμνει

το P πάνω από το b ή η απόσταση του b από την πλευρά C είναι μικρότερη από αυτή της κορυφής a-1 τότε κάνουμε “flush” την πλευρά B με το P και ενημερώνουμε το beta ανάλογα. Στη συνέχεια αν η απόσταση του μέσου της ακμής [b,b-1] από την C είναι μικρότερη από αυτή της κορυφής a-1 τότε η τρίτη πλευρά πρέπει να περνάει και να έχει μέσο την κορυφή a-1. Τέλος, επιστρέφουμε τα alpha(δείκτης), mid_alpha(δείκτης), b(κορυφή), beta(λίστα με τις δύο κορυφές από τις οποίες περνάει η πλευρά B), yb(σημείο), flush_beta(δείκτης).

3.3.10 triangle_points()

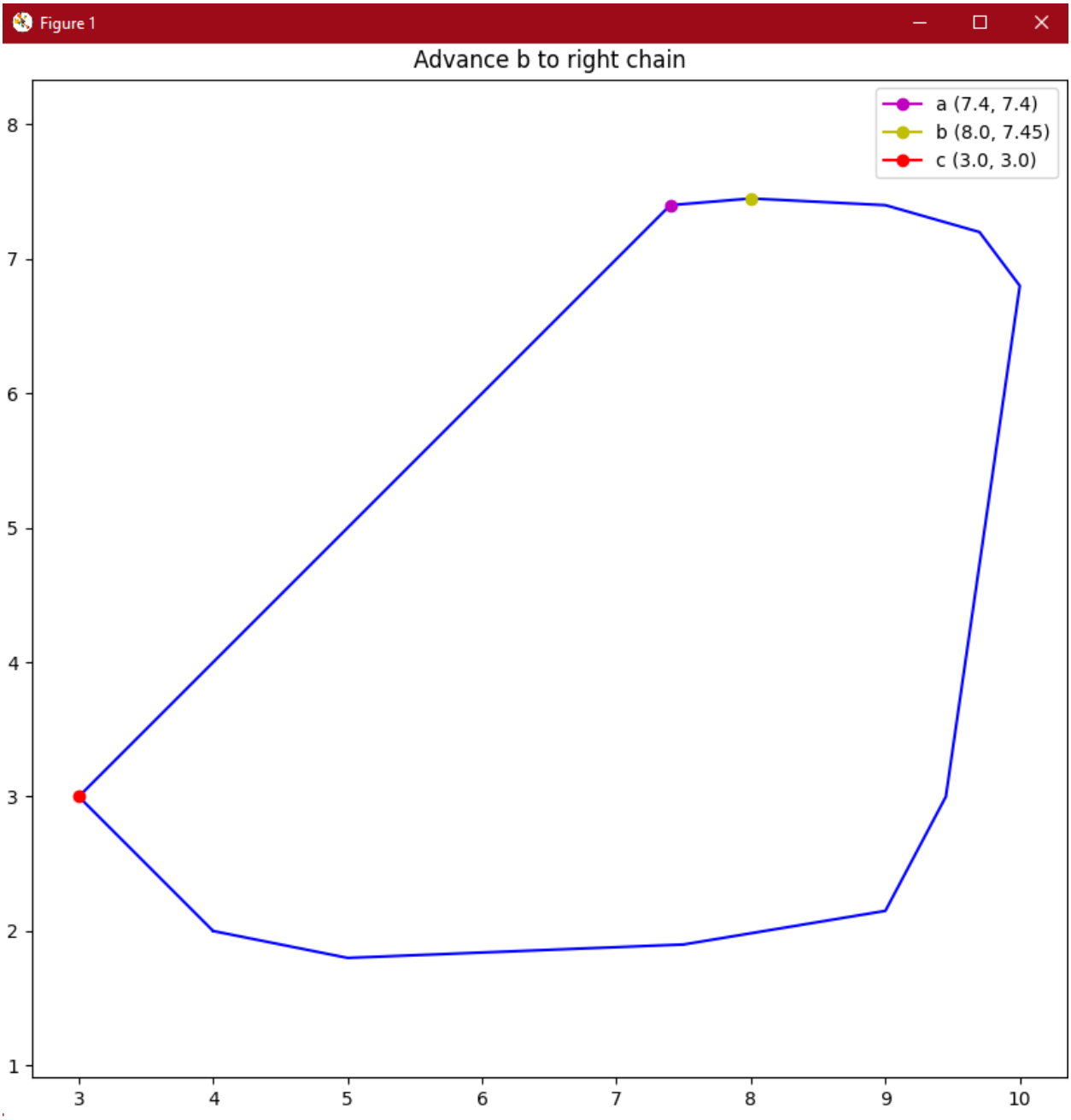
Υπολογίζει και επιστρέφει τα σημεία του ελάχιστου περιεγραμμένου τριγώνου. Αρχικά το B είναι η τομή του beta με την πλευρά C. Ανάλογα με τον δείκτη mid_alpha υπολογίζουμε και τα άλλα δύο σημεία. Για την περίπτωση που θέλουμε να έχει συγκεκριμένο μέσο, υπολογίζουμε το απέναντι διαγώνιο σημείο του B σε σχέση με το μέσο και στη συνέχεια υπολογίζουμε το alpha, χρησιμοποιώντας την παράλληλη γραμμή της απέναντι πλευράς που περνάει από αυτό το σημείο. Το alpha είναι το σημείο τομής της παράλληλης με την πλευρά B.

3.3.11 draw_triangle()

Σχεδιάζει το τρίγωνο, υπολογίζει και επιστρέφει το εμβαδόν του.

3.3.12 Παράδειγμα Εκτέλεσης του Αλγόριθμου

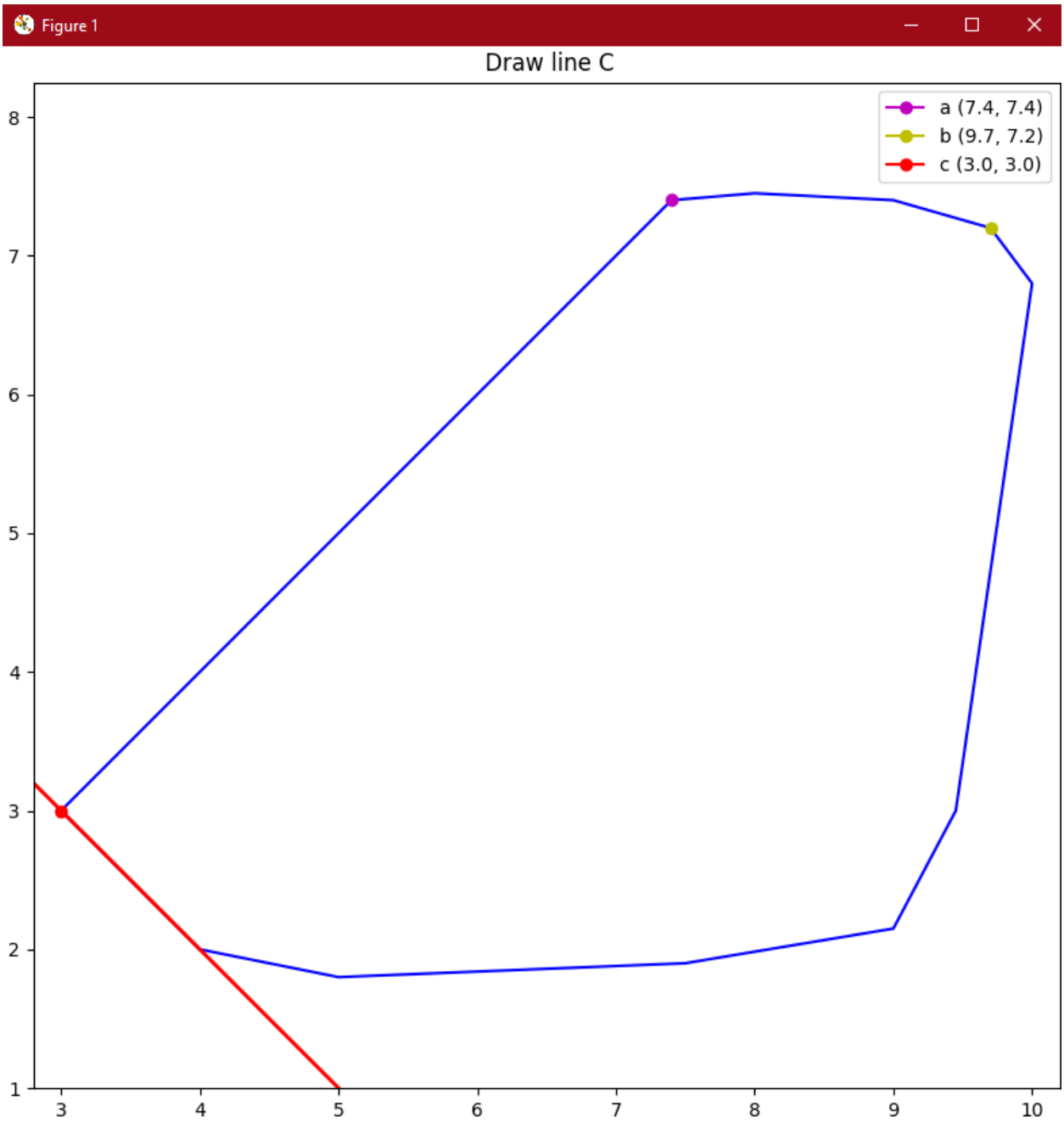
- Η αρχική κατάσταση του αλγορίθμου φαίνεται στο Σχήμα 3.3. Όπως μπορούμε να παρατηρήσουμε τα a, b και c αντιστοιχούν στη δεύτερη, τρίτη και πρώτη ωρολογιακά κορυφή του πολυγώνου. Βρισκόμαστε στο εσωτερικό της πρώτης for, όπου με τη βοήθεια των clear_plot και animation_step μας έχει εμφανιστεί το κατάλληλο σχήμα. Το επόμενο βήμα του αλγορίθμου φαίνεται στον τίτλο του κάθε σχήματος.



Σχήμα 3.3

– Όπως μας ενημέρωσε ο τίτλος του προηγούμενου σχήματος (Σχήμα 3.3), βρισκόμαστε στο advance_b_to_right_chain() το αποτέλεσμα του οποίου μπορούμε να δούμε στο Σχήμα 3.4. Συγκεκριμένα, παρατηρούμε το b να έχει "προχωρήσει" δύο κορυφές σε ωρολογιακή πάντα φορά και να βρίσκεται πλέον σε δεξιά αλυσίδα. Μια ακόμα παρατήρηση που μπορούμε να κάνουμε, η οποία όμως δεν σχετίζεται με την περάτωση του αλγορίθμου, είναι η κόκκινη γραμμή η οποία είναι

η μια από τις πλευρές του περιγεγραμμένου τριγώνου και πιο συγκεκριμένα λόγο της σύμβασης που έχουμε κάνει θα είναι πάντα flush με το P και είναι η πλευρά C.



Σχήμα 3.4

– Αφού έχουμε υπολογίσει το γ_a το εμφανίζουμε στο Σχήμα 3.5.

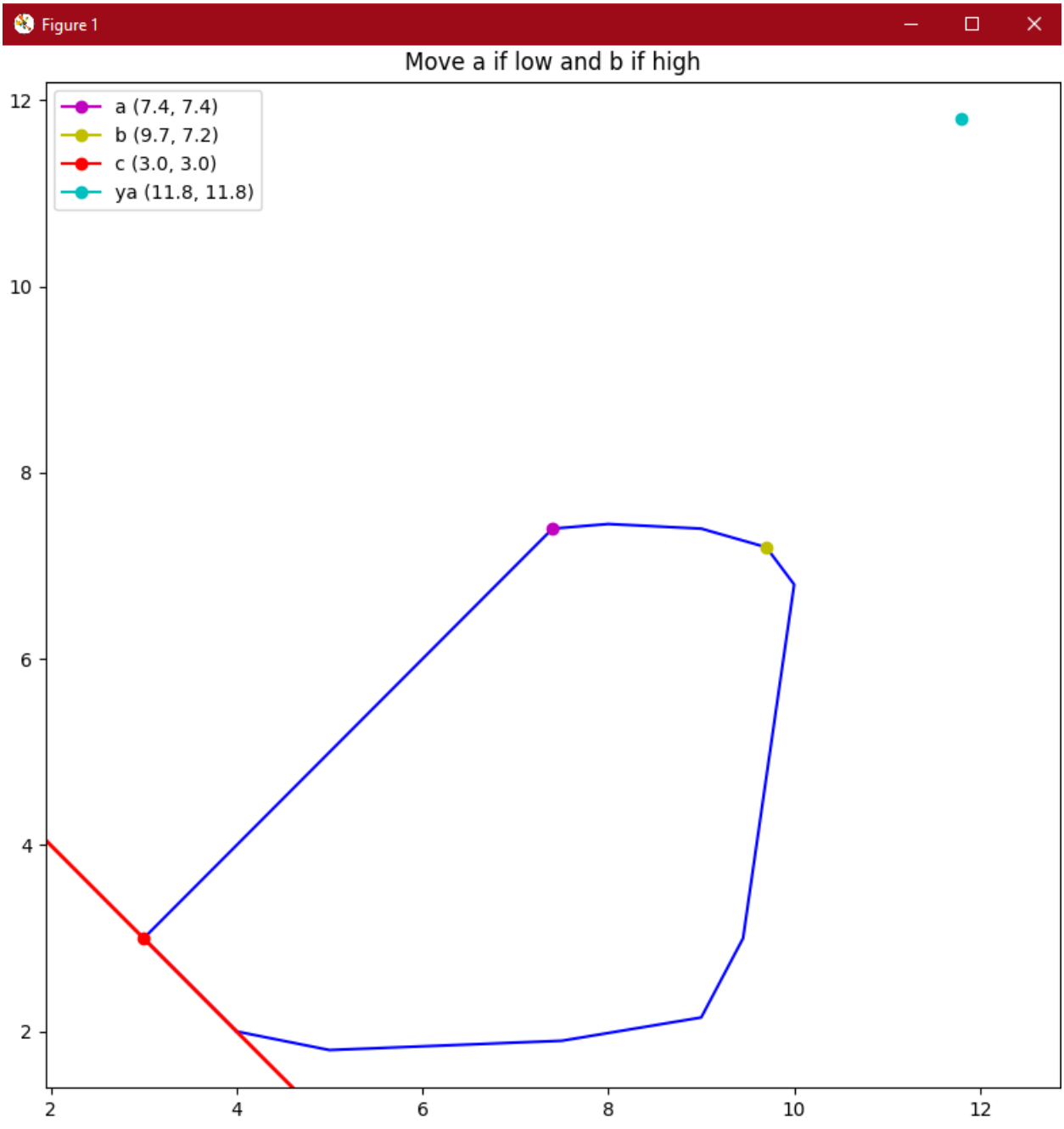
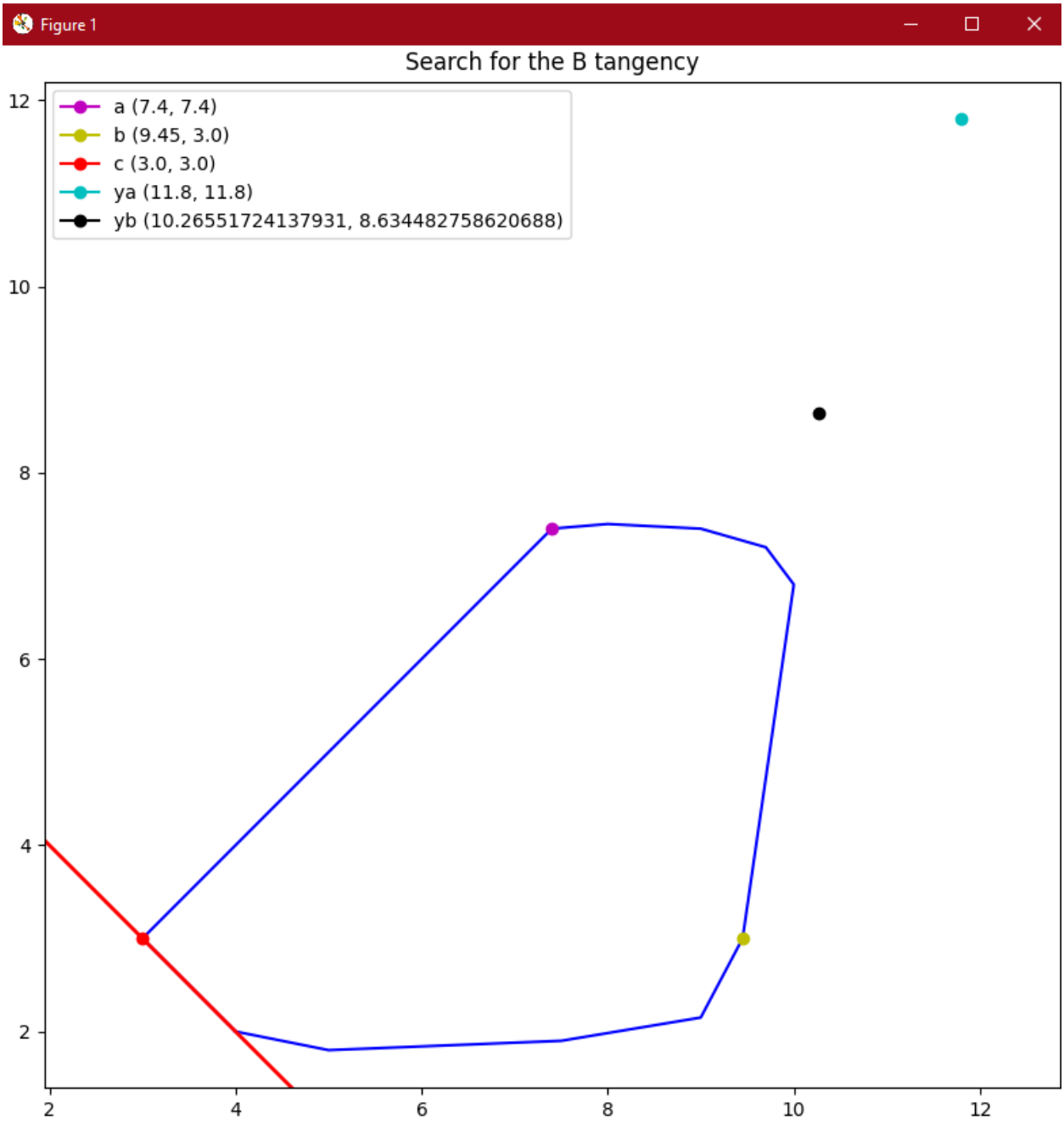


Figure 1
Move a if low and b if high
a (7.4, 7.4)
b (9.7, 7.2)
c (3.0, 3.0)
ya (11.8, 11.8)
12
10
8
6
4
2
2
4
6
8
10
12

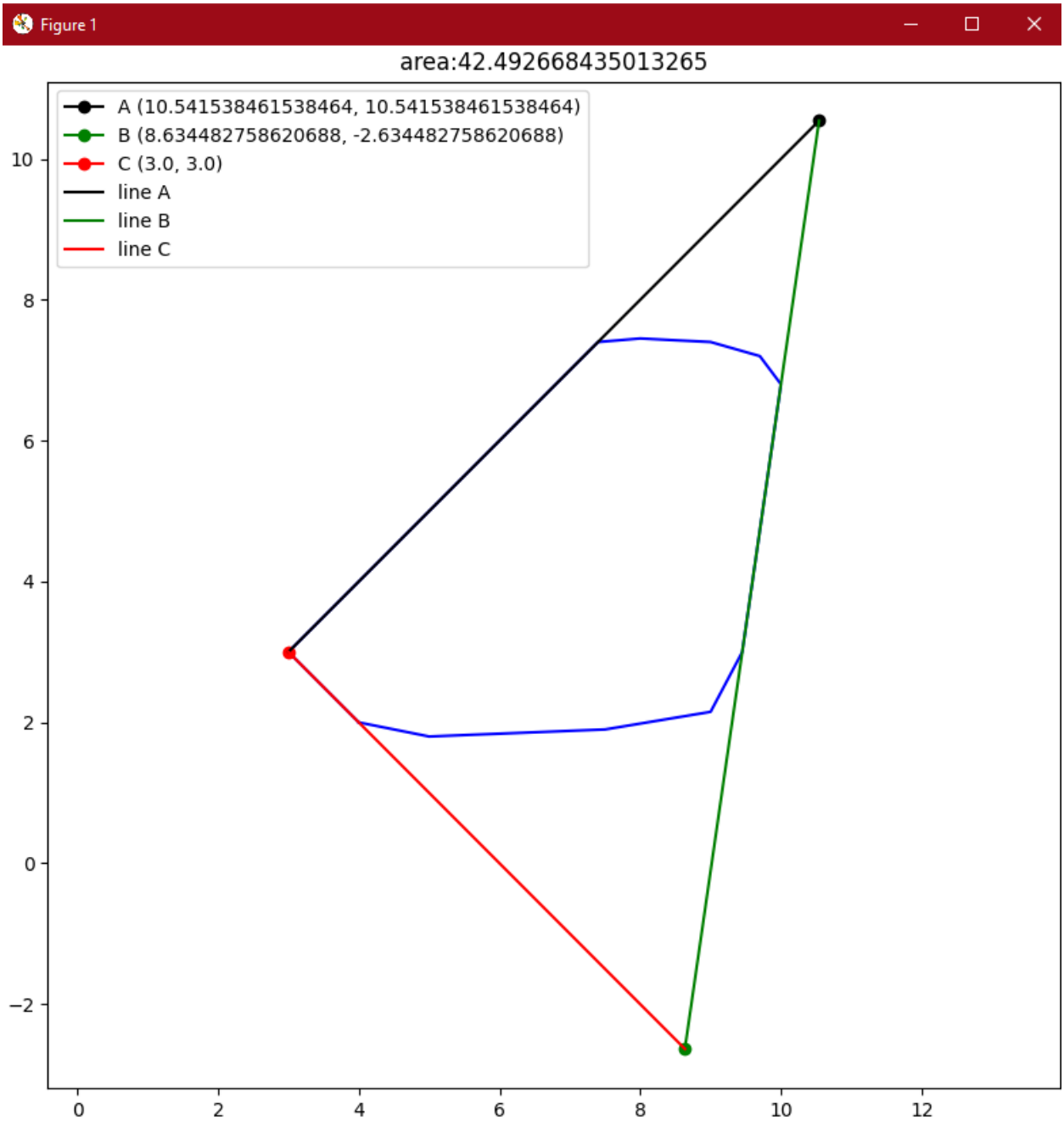
Σχήμα 3.5

– Όπως παρατηρούμε στο Σχήμα 3.6 το a δεν ήταν χαμηλό, οπότε δεν αλλάζει θέση. Επίσης έχουμε υπολογίσει το γ_b, επίσης έχουμε εκτελέσει τον κώδικα για του search_for_B_tangency() με αποτέλεσμα το b να έχει μετακινηθεί πιο δεξιά, διότι η ευθεία γ_bb έτεμνε το P κάτω από το b.



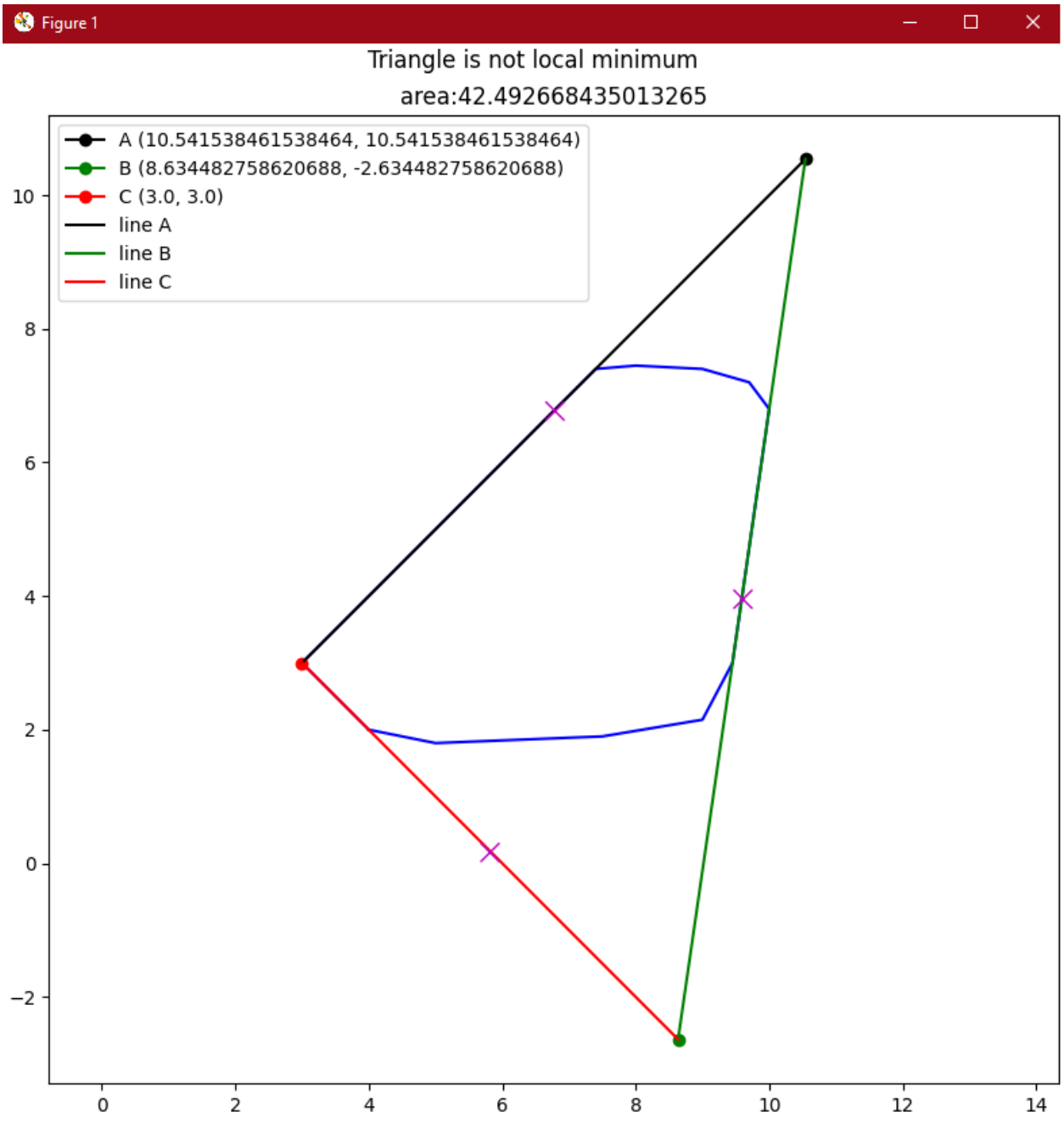
Σχήμα 3.6

– Η triangle_points() υπολογίζει τα υπόλοιπα δύο σημεία του περιγεγραμμένου τριγώνου, η triangle_area() υπολογίζει το εμβαδόν, το οποίο η draw_triangle() το εμφανίζει στο τίτλο του Σχήματος 3.7 καθώς επίσης ενώνει τις τρεις κορυφές για να εμφανιστεί το τρίγωνο.



Σχήμα 3.7

– Τέλος η local_minimum() υπολογίζει τα μέσα κάθε πλευράς και αν εφάπτονται με το P τότε το τρίγωνο είναι τοπικό ελάχιστο, αλλιώς όχι. Όπως μπορούμε να παρατηρήσουμε στο Σχήμα 3.8 το περιγεγραμμένο τρίγωνο που υπολογίζαμε δεν είναι τοπικό ελάχιστο.



Σχήμα 3.8

3.4 Υλοποίηση Αλγόριθμου 2

3.4.1 maximal_enclosed_triangle ()

```
def maximal_enclosed_triangle(
    a, b, c, alpha, beta, gamma, mid_alpha, flush_b, n, global_maximum, polygon
):
    # first inner triangle
    # ypologizetai pairnontas tin epomeni clockwise koryfi apo ta mid points twn
    # [a,a-1] kai [b,b-1] i opoia einai i a kai b antistoixa
    print("first inner triangle")
    first_inner_a = check_chord_parallel(a, b, c, n, polygon)

    first_inner_b = check_chord_parallel(b, c, a, n, polygon)

    first_inner_c = polygon[c]

    inner_triangle_area = draw_triangle(
        first_inner_a, first_inner_b, first_inner_c, polygon
    )

    global_maximum = global_maximum_area(
        first_inner_a, first_inner_b, polygon[c], global_maximum, inner_triangle_area
    )

    # elegxoume an to mid_alpha kai to flush_b kai an einai 0 kai 1 antistoixa, to
trigwno exei 2 flush legs
    # afoy i pleura A de xriazetai na pernaei apo to mid_alpha, epomenws einai flush
me to [a,a-1] kai
    # i pleura B einai flush me to [b,b-1], kathos tin exoume orisei etsi
    if not mid_alpha and flush_b:
        # ---------------------------two flush legs--------------------------
        print("2 flush legs")
        # flag gia to case '2 flush legs'
        case = 1
        x = midpoint(alpha, polygon[c])
        print("x:", x)
        x_x, x_y = x
        c_line_x, c_line_y = line(c, polygon)
        plt.plot(x_x, x_y, marker="X", color="r", label=f"x {x_x, x_y}")
        plt.axline(c_line_x, c_line_y, linewidth=1, color="m")
        plt.axline(alpha, gamma, linewidth=1, color="m")
        plt.axline(beta, alpha, linewidth=1, color="m")
        (
            inner_triangle_area,
            inner_a,
            inner_b,
        ) = two_flush_legs(a, b, c, alpha, beta, gamma, x, case, n, polygon)
    else:
        print("1 flush leg")
        # flag gia to case '1 flush leg'
        case = 0
        (
            inner_triangle_area,
            inner_a,
            inner_b,
        ) = two_flush_legs(
```

```
            a, b, c, alpha, beta, gamma, polygon[a - 1], case, n, polygon
        )
        print(alpha, beta, gamma)
        c_line_x, c_line_y = line(c, polygon)
        plt.axline(c_line_x, c_line_y, linewidth=1, color="m")
        plt.axline(alpha, gamma, linewidth=1, color="m")
        plt.axline(beta, alpha, linewidth=1, color="m")
    global_maximum = global_maximum_area(
        inner_a,
        inner_b,
        polygon[c],
        global_maximum,
        inner_triangle_area,
    )
    return global_maximum
```

```
def two_flush_legs(a, b, c, alpha, beta, gamma, x, case, n, polygon):
    # calculate theta angles
    theta_a, theta_b, theta_c = theta_angle(
        a, b, c, x, alpha, beta, gamma, case, n, polygon
    )

    # a(theta) = a'
    inner_triangle_area_theta_a = inner_triangle_base_of_theta(
        a, b, c, alpha, beta, gamma, x, theta_a, case, n, polygon, "a"
    )

    # b(theta) = b'
    inner_triangle_area_theta_b = inner_triangle_base_of_theta(
        a, b, c, alpha, beta, gamma, x, theta_b, case, n, polygon, "b"
    )

    min_theta = min(theta_a, theta_b, theta_c)
    inner_triangle_area_theta_min = inner_triangle_base_of_theta(
        a, b, c, alpha, beta, gamma, x, min_theta, case, n, polygon, "min_theta"
    )

    # epistrefoume to megisto trigwno apo ta parapanw
    inner_triangle_area, inner_a, inner_b = max(
        inner_triangle_area_theta_min,
        inner_triangle_area_theta_a,
        inner_triangle_area_theta_b,
    )
    return inner_triangle_area, inner_a, inner_b
```

```
def inner_triangle_base_of_theta(
    a, b, c, alpha, beta, gamma, x, theta, case, n, polygon, name
):
    # to C_star einai i tomi tis pleuras A me tin nea C pou einai stramenei kata angle
clockwize
    # to triangle einai mia lista tis morfis (area, vertex_a, vertex_b, vertex_c)
    if name == "a" or name == "b":
        # gia na elegksoume mono to theta pou exei kalesei tin function
        # apo to locals theloume to name to opoio tha einai a i b
        # an theloume to theta_a tote to opposite vertex tha einai to b kai anapoda
        name_value = locals()[name]
        opposite_vertex = locals()["b"] if name == "a" else locals()["a"]

        triangle = validation(
```

```
            polygon[name_value],
            x,
            polygon[opposite_vertex],
            polygon[opposite_vertex - 1],
            c,
            name,
            polygon,
        )

        triangle2 = validation(
            polygon[name_value],
            rotate(polygon[name_value], x, theta),
            polygon[opposite_vertex],
            polygon[opposite_vertex - 1],
            c,
            name,
            polygon,
        )

        plt.waitforbuttonpress()
        return max(triangle, triangle2)
    else:
        C_star_validation = [
            alpha,
            polygon[a - 1],
            polygon[c],
            rotate(polygon[c], polygon[c - 1], theta),
        ]
        if not valid(*C_star_validation):
            return [-1, None, None]

        C_star = intersecting_lines(*C_star_validation)
        print("C_star: ", C_star)
        C_star_A_midpoint = midpoint(C_star, alpha)

        B_star_validation = [
            polygon[b],
            polygon[b - 1],
            polygon[c],
            rotate(polygon[c], polygon[c - 1], theta),
        ]
        if not valid(*B_star_validation):
            return [-1, None, None]
        # to B_star einai i tomi tis pleuras B me tin nea C pou einai stramenei kata
angle clockwize
        B_star = intersecting_lines(*B_star_validation)

        B_star_A_midpoint = midpoint(B_star, alpha)

        print(f"theta_{name}")
        area = draw_triangle(C_star_A_midpoint, B_star_A_midpoint, polygon[c],
polygon)
        triangle = [area, C_star_A_midpoint, B_star_A_midpoint]
    return triangle
```

Η maximal_enclosed_triangle() είναι υπεύθυνη για το υπολογισμό του μέγιστου εγγεγραμμένου τριγώνου. Σαν είσοδο παίρνει τις κορυφές στις οποίες βρισκόμαστε στο πολύγωνο, τις κορυφές του ελάχιστου περιγεγραμμένου τριγώνου που βρήκαμε στον προηγούμενο αλγόριθμο και τους δείκτες για να γνωρίζουμε σε ποιες περιπτώσεις είμαστε. Το πρώτο τρίγωνο υπολογίζεται παίρνοντας την επόμενη ωρολογιακά

κορυφή από τα μέσα των ακμών του περιγεγραμμένου τριγώνου. Κάνοντας δύο ελέγχους, σε σταθερό χρόνο, μπορούμε να αποδεχτούμε το τρίγωνο ή να αυξήσουμε το εμβαδόν του. Αναλυτικότερα, υπολογίζουμε αν η παράλληλη με το chord που περνάει από το τρίτο σημείο τέμνει το P. Εάν το τέμνει τότε το τρίγωνο μπορεί να "μεγαλώσει", αυξάνοντας το τρίτο σημείο κατά μια θέση ωρολογιακά. Το συγκεκριμένο έλεγχο τον πραγματοποιούμε για τα δύο chords που σχηματίζονται από την κορυφή c, η οποία είναι η πιο ακραία ωρολογιακά κορυφή της πλευράς C, προς τις άλλες δύο κορυφές του εγγεγραμμένου τριγώνου. Τα επόμενα τρίγωνα αναλαμβάνει να τα υπολογίσει η two_flush_legs, η οποία ανάλογα σε ποια περίπτωση βρισκόμαστε κάνει τους κατάλληλους υπολογισμούς. Τέλος επιστρέφει το ολικό μέγιστο τρίγωνο με τη μορφή λίστας εμβαδού, συντεταγμένων. Αναλυτικότερα στο εσωτερικό της βρίσκονται οι εξής συναρτήσεις:

3.4.2 check_chord_parallel ()

Ελέγχουμε αν το εμβαδόν του αρχικού εσωτερικού τριγώνου μπορεί να αυξηθεί, παίρνοντας ως ορίσματα την κορυφή που θέλουμε να ελέγξουμε, τις κορυφές του chord σε ωρολογιακή φορά και το πλήθος των ακμών του P. Επιστρέφουμε την επόμενη ωρολογιακά κορυφή, εάν το εμβαδόν μπορεί να αυξηθεί αλλιώς την ίδια.

3.4.3 global_maximum_area ()

Σαν ορίσματα παίρνουμε τις κορυφές, το μέγιστο μέχρι εκείνη τη στιγμή τρίγωνο και το εμβαδόν που υπολογίσαμε. Ελέγχουμε αν το εμβαδόν του τριγώνου είναι το μέγιστο και αν είναι επιστρέφουμε το τρίγωνο αυτό σαν μια λίστα της μορφής [εμβαδόν, κορυφή α, κορυφή β, κορυφή γ] αλλιώς επιστρέφουμε το προηγούμενο μέγιστο.

3.4.4 two_flush_legs ()

Ανάλογα με την περίπτωση στην οποία βρισκόμαστε, η two_flush_legs υλοποιείται διαφορετικά. Αναλυτικότερα:

- Ελέγχουμε το flag mid_alpha και το flush_b, εάν είναι μηδέν και ένα αντίστοιχα, τότε το περιγεγραμμένο τρίγωνο έχει δύο flush legs, αφού η πλευρά A δεν περνάει από το mid_alpha, επομένως είναι flush με την ακμή [a, a-1] και η πλευρά B αντίστοιχα είναι flush με την ακμή [b, b-1]. Στη συνέχεια καλούμε την theta_angle () η οποία υπολογίζει τις γωνίες θ_a θ_b θ_c του Σχήματος 10.

- Στην αντίθετη περίπτωση, οι γωνίες θ_a θ_b θ_c που υπολογίζουμε είναι του Σχήματος 11.

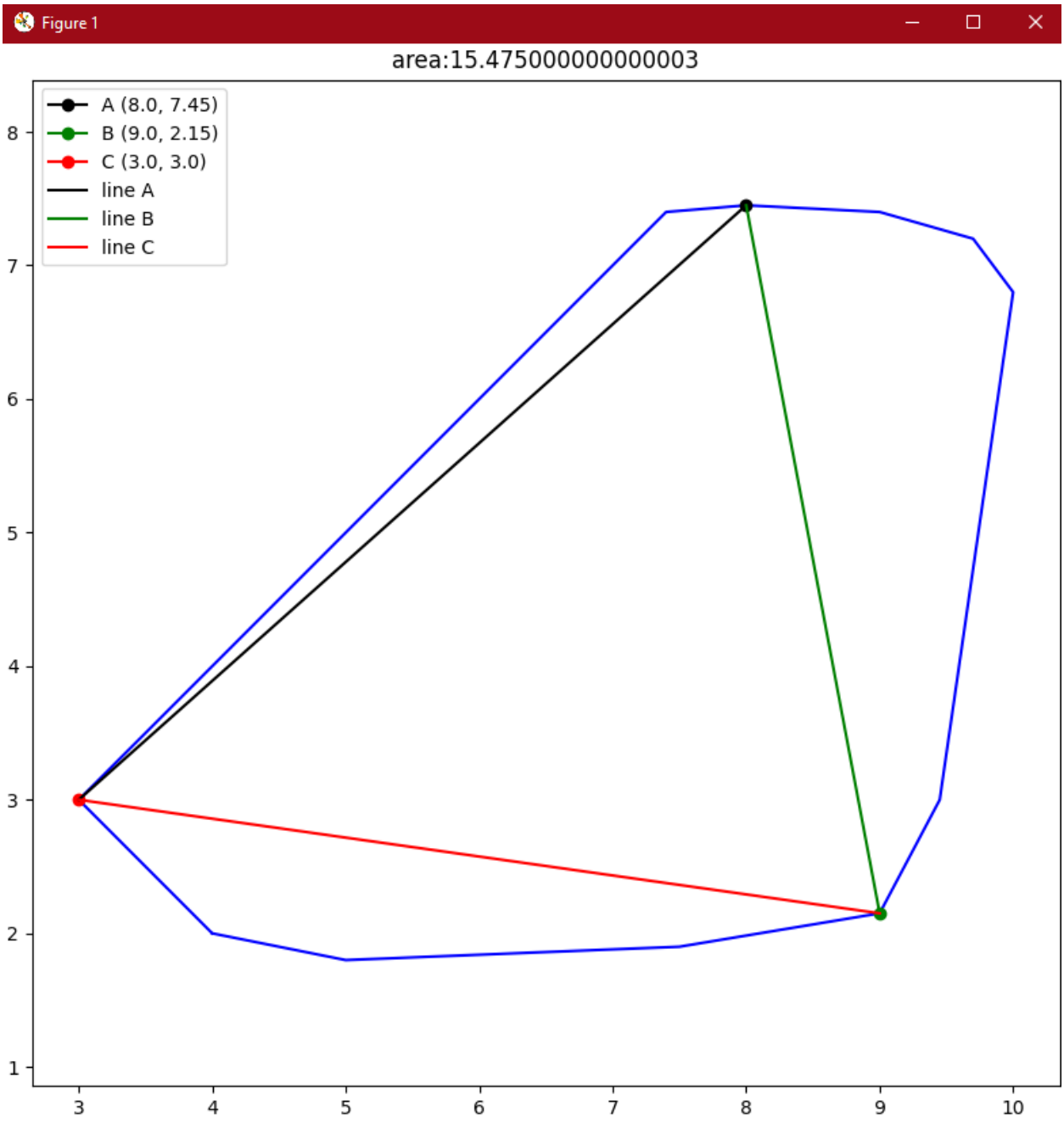
Τέλος επιστρέφει το μέγιστο δυνατό εσωτερικό τρίγωνο, με βάση τις γωνίες που υπολογίσαμε καθώς και τις κορυφές του.

3.4.5 inner_triangle_base_of_theta()

Ανάλογα με τις γωνίες που βρέθηκαν στην two_flush_legs() η inner_triangle_base_of_theta() επιστρέφει το μέγιστο δυνατό τρίγωνο αφού πρώτα έχει περάσει από validation το οποίο μας εξασφαλίζει το τρίγωνο να είναι εξ ολοκλήρου εσωτερικά του P.

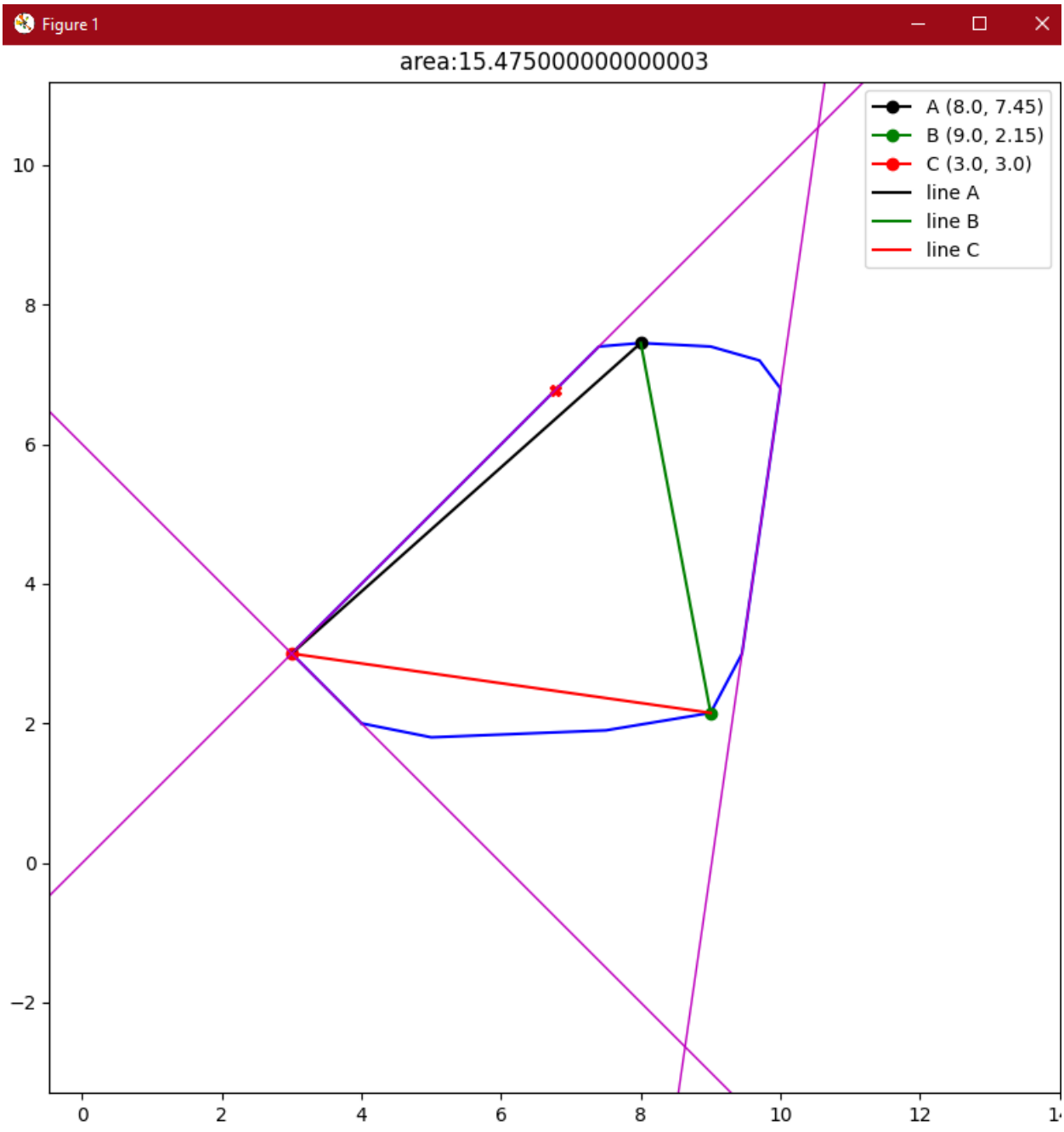
3.4.6 Παράδειγμα Εκτέλεσης του Αλγόριθμου

- Η αρχική κατάσταση του αλγορίθμου φαίνεται στο Σχήμα 3.9. Όπως μπορούμε να παρατηρήσουμε τα a, b βρίσκονται μια θέση πιο δεξιά, στις κορυφές του πολυγώνου από τις οποίες σταμάτησε ο προηγούμενος αλγόριθμος ενώ η c παραμένει σταθερή. Ο λόγος της μεταβολής αυτής, είναι το αποτέλεσμα της check_chord_parallel() η οποία "μεγάλωσε" το τρίγωνο όπως περιγράφεται στην ενότητα 3.4.2. Η draw_triangle() εμφάνισε το τρίγωνο και με την global_maximum_area() ελέγχουμε αν είναι το μεγαλύτερο από όσα έχουμε βρει.



Σχήμα 3.9

– Στη συνέχεια και όπως μπορούμε να δούμε από το Σχήμα 3.10, βρισκόμαστε στην κατηγορία two flush legs. Υπολογίζουμε το x και καλούμε την two_flush_legs().



Σχήμα 3.10

– Η two_flush_legs() με τη σειρά της υπολογίζει τα θ_a θ_b θ_c του Σχήματος 10 μέσω της theta_angle(). Τα θ_a, θ_b και θ_c είναι 180^o , 170.39^o και 0^o αντίστοιχα. Από τις συγκεκριμένες γωνίες και έπειτα από την validation() η inner_triangle_base_of_theta() εμφάνισε μόνο το εσωτερικό τρίγωνο που σχηματίστηκε από τη γωνία θ_c (Σχήμα 3.11), διότι ήταν το μόνο που πληρούσε τις προϋποθέσεις.

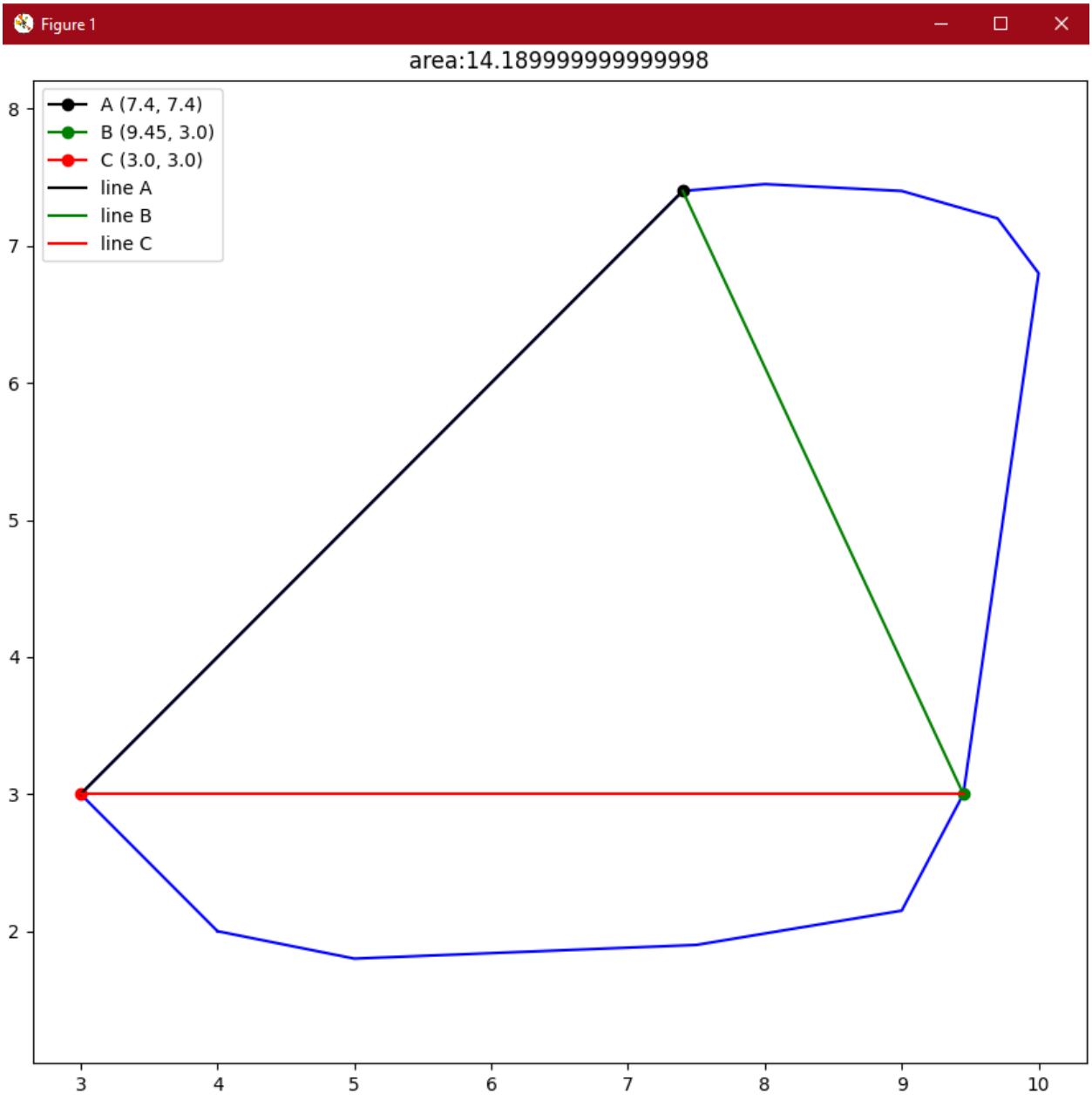


Figure 1
area:14.189999999999998
A (7.4, 7.4)
B (9.45, 3.0)
C (3.0, 3.0)
line A
line B
line C
8
7
6
5
4
3
2
3
4
5
6
7
8
9
10

Σχήμα 3.11

4

Επίλογος

Στόχος της συγκεκριμένης διπλωματικής εργασίας ήταν η μελέτη και η υλοποίηση δύο αλγορίθμων το αποτέλεσμα των οποίων έχει μια πληθώρα εφαρμογών. Ενώ φαινομενικά ήταν δύο ξένα μεταξύ τους προβλήματα, όπως και είχαν αντιμετωπιστεί στα αρχικά στάδια μελέτης τους, καταλήξαμε σε έναν ενιαίο αλγόριθμο ο οποίος χρησιμοποιεί γνώσεις του πρώτου για την επίλυση και του δεύτερου προβλήματος.

Ο πρώτος ήταν ένας αλγόριθμος όπου υπολόγιζε το τρίγωνο με το μικρότερο εμβαδόν που είναι περιγεγραμμένο σε ένα δοθέν κυρτό πολύγωνο. Μάλιστα ο συγκεκριμένος ήταν και η βέλτιστη δυνατή λύση στο πρόβλημα αυτό. Ο επόμενος ήταν ένας ακολουθιακός αλγόριθμος, ο οποίος αξιοποιεί το συμπέρασμα του πρώτου, με σκοπό την εύρεση του μέγιστου δυνατού τριγώνου το οποίο είναι εγγεγραμμένο, αυτή τη φορά, στο κυρτό πολύγωνο. Και τέλος ένας ακόμα στόχος της διπλωματικής εργασίας ήταν η υλοποίηση ενός εργαλείου, όπως πολλάκις έχει αποδειχθεί, το οποίο ήταν η οπτικοποίηση αυτών.

Βιβλιογραφία

1. Joseph O'Rourke, Alok Aggarwal, Sanjeev Maddila, and Michael Baldwin, An Optimal Algorithm for Finding Minimal Enclosing Triangles, JOURNAL OF ALGORITHMS 7, pp. 258-269, 1986.
2. Victor Klee and Michael C. Laskowski Finding the Smallest Triangles Containing a Given Convex Polygon, JOURNAL OF ALGORITHMS 6, pp. 359-375, 1985.
3. Godfried T. Toussaint, Rotating calipers , Proceedings of IEEE MELECON'83, Athens, Greece, May 1983 https://en.wikipedia.org/wiki/Rotating_calipers
4. J. E. Boyce, D. P. Dobkin, R. L. Drysdale, and L.J. Guibas, Finding extremal polygons, SIAM Journal on computing 14, pp. 134-147, 1985.
5. A. Aggarwal, J. K. Park, Sequential searching in multidimensional monotone arrays, 29th Annual IEEE Symposium on Foundations of Computer Science, pp. 497-512, 1988
6. D. P. Dobkin and L. Snyder, On a general method for maximizing and minimizing among certain geometric problems, 20th Annual IEEE Symposium on Foundations of Computer Science, pp. 9-17, 1979
7. Sharat Chandran, and David M. Mount, A Parallel Algorithm for Enclosed and Enclosing Triangles, International Journal of Computational Geometry & Applications 2 (2), pp. 191-214, 1992