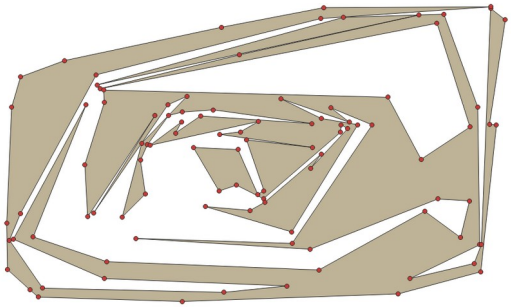
Παρουσίαση αποτελεσμάτων για τον αλγόριθμο Simulated Annealing

Παρουσιάζουμε αποτελέσματα μόνο για το local annealing, καθώς το global annealing δεν έχει πάντα σωστά αποτελέσματα, και το subdivision δεν έχει υλοποιηθεί.

Αυξάνοντας τον αριθμό των σημείων

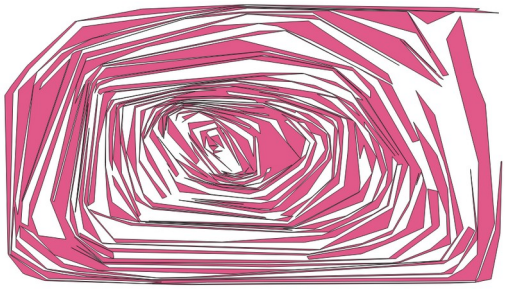
Εντολή:
./optimal_polygon -i ../instances/data/images/euro-night-0000100.instance -o result.txt -algorithm simulated_annealing -annealing local -L 1000 -max -gen onion

Αποτέλεσμα:
construction time: 180ms



Εντολή:
./optimal_polygon -i ../instances/data/images/euro-night-0001000.instance -o result.txt -algorithm simulated_annealing -annealing local -L 1000 -max -gen onion

Αποτέλεσμα:
construction time: 2398ms



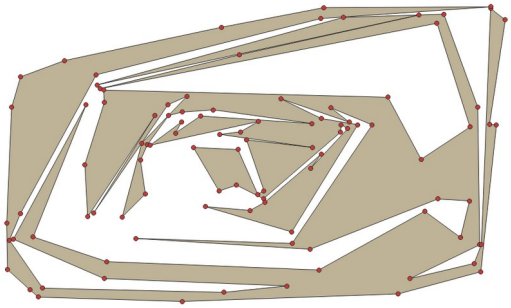
Εντολή:
./optimal_polygon -i ../instances/data/images/euro-night-0010000.instance -o result.txt -algorithm simulated_annealing -annealing local -L 1000 -max -gen onion

Αποτέλεσμα:
construction time: 49035ms

Σχόλια:
Όπως περιμέναμε αυξάνοντας τον αριθμό των σημείων αυξάνεται ο χρόνος κατασκευής του βέλτιστου πολύγωνου. Όμως αυξάνεται ικανοποιητικά. 0.2 sec για 100 σημεία, 2.4 sec για 1000 σημεία και 49 sec για 10000 σημεία.

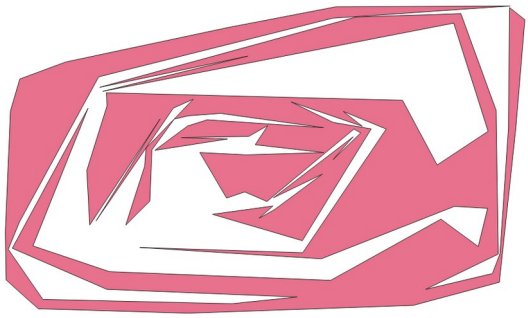
Αυξάνοντας τον αριθμό των επαναλήψεων του αλγορίθμου (παράμετρος L)
Εντολή:
./optimal_polygon -i ../instances/data/images/euro-night-0000100.instance -o result.txt -algorithm simulated_annealing -annealing local -L 1000 -max -gen onion

Αποτέλεσμα:
construction time: 180ms



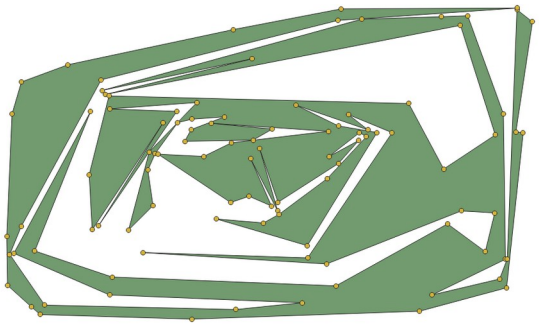
Εντολή:
./optimal_polygon -i ../instances/data/images/euro-night-0000100.instance -o result.txt -algorithm simulated_annealing -annealing local -L 5000 -max -gen onion

Αποτέλεσμα:
construction time: 906ms



Εντολή:
./optimal_polygon -i ../instances/data/images/euro-night-0000100.instance -o result.txt -algorithm simulated_annealing -annealing local -L 10000 -max -gen onion

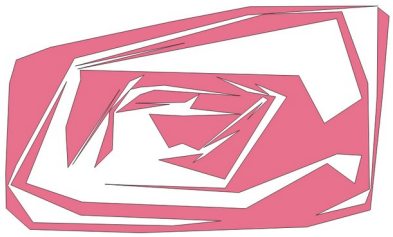
Αποτέλεσμα:
construction time: 1710ms



Σχόλια:
Όπως περιμέναμε αυξάνεται ο χρόνος κατασκευής του βέλτιστου πολυγώνου, αλλά βελτιώνεται και το αποτέλεσμα. Αυτό είναι λογικό, όσες περισσότερες επαναλήψεις κάνει ο αλγόριθμος τόσο περισσότερο βελτιώνεται το πολύγωνο.

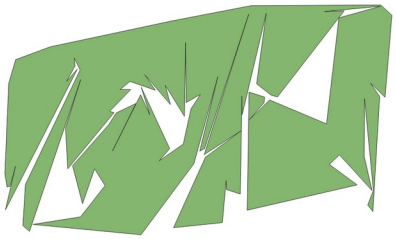
Αλλάζοντας αλγόριθμο κατασκευής (μεγιστοποίηση)
Εντολή: ./optimal_polygon -i ../instances/data/images/euro-night-0000100.instance -o result.txt -algorithm simulated_annealing -annealing local -L 5000 -max -gen onion

Αποτέλεσμα:
construction time: 906ms
area: 4.15413e+07



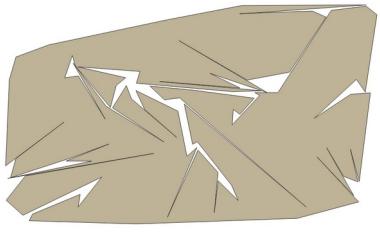
Εντολή: ./optimal_polygon -i ../instances/data/images/euro-night-0000100.instance -o result.txt -algorithm simulated_annealing -annealing local -L 5000 -max -gen incremental

Αποτέλεσμα:
construction time: 718ms
area: 5.90613e+07



Εντολή: ./optimal_polygon -i ../instances/data/images/euro-night-0000100.instance -o result.txt -algorithm simulated_annealing -annealing local -L 5000 -max -gen convex_hull

Αποτέλεσμα:
construction time: 787ms
area: 6.93323e+07



Σχόλια: Βλέπουμε ότι ανεξάρτητα από τον αλγόριθμο κατασκευής του αρχικού πολυγώνου έχουμε παραπλήσιους χρόνους βελτιστοποίησης. Όμως το τελικό αποτέλεσμα είναι καλύτερο αν κατασκευάσουμε το αρχικό πολύγωνο με τον Convex Hull, από ότι με τον Incremental και τα δύο καλύτερα με από τον Onion. Αυτό επειδή τα πολύγωνα που κατασκευάζουν οι αλγόριθμοι αυτοί έχουν από την αρχή την ίδια διάταξη ως προς το εμβαδόν του πολυγώνου που κατασκευάζουν. Δηλαδή αν θέλουμε μεγιστοποίηση, μας βοηθάει να επιλέξουμε τον Incremental, γιατί κατασκευάζει πολύγωνο μεγαλύτερου εμβαδού από ότι ο Onion, και είναι σημαντικά ταχύτερος του

Convex Hull αν και ο Convex Hull έχει ακόμη μεγαλύτερο εμβαδόν στο πολύγωνο που κατασκευάζει. Δηλαδή αν θέλουμε μεγιστοποίηση και δεν μας ενδιαφέρει ο χρόνος πάμε με Convex Hull, αλλιώς αν θέλουμε και καλό αποτέλεσμα αλλά και σε καλό χρόνο πάμε με Incremental. Για ελαχιστοποίηση θα πάμε σε κάθε περίπτωση με τον Onion, αφού είναι ο ταχύτερος στην κατασκευή αρχικού πολυγώνου και το πολύγωνο που παράγει έχει το μικρότερο εμβαδόν σε σχέση με τους άλλους δύο αλγόριθμους.