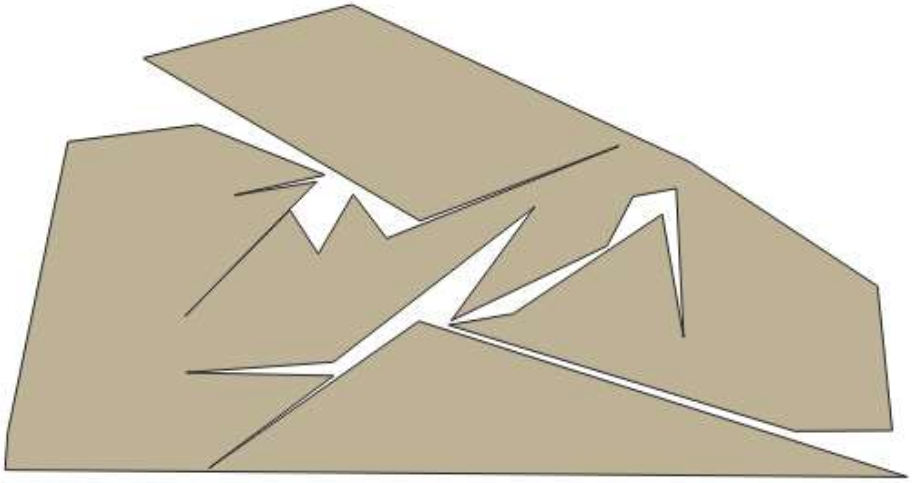
Σημείωση: Στην μεγιστοποίηση σκοπός μας είναι να ελαττώσουμε όσο είναι δυνατόν το αρχικό ratio. Στην ελαχιστοποίηση σκοπός μας είναι να μεγαλώσουμε όσο είναι δυνατό το αρχικό ratio

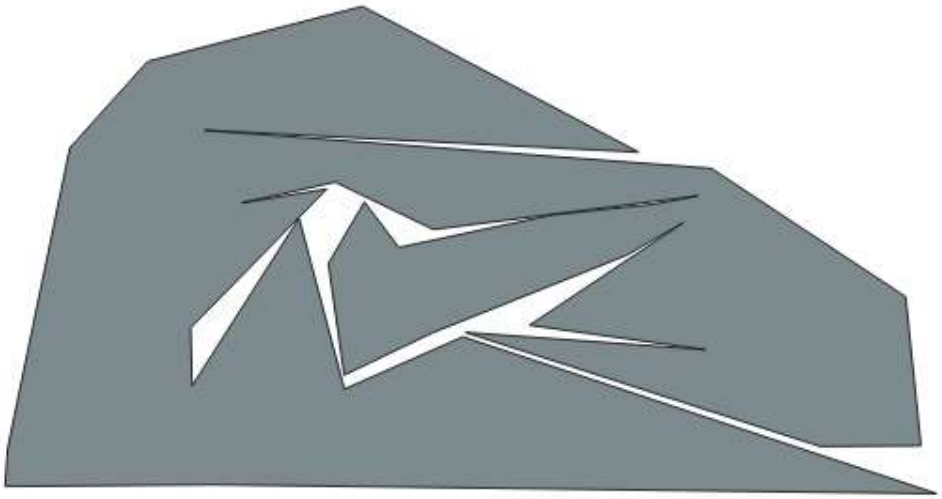
Όπως αναφέρθηκε και στο README, η επιλογή του αλγορίθμου για την δημιουργία και την αρχικοποίηση του αρχικού πολυγώνου επηρεάζει σημαντικά όχι μόνο τον χρόνο εκτέλεσης αλλά και το τελικό εμβαδό που προκύπτει.

Μελετώντας τα αποτελέσματα για το σύνολο **euro-night-35** για **μεγιστοποίηση**, με L=10,threshold=0.95 έχουμε ότι

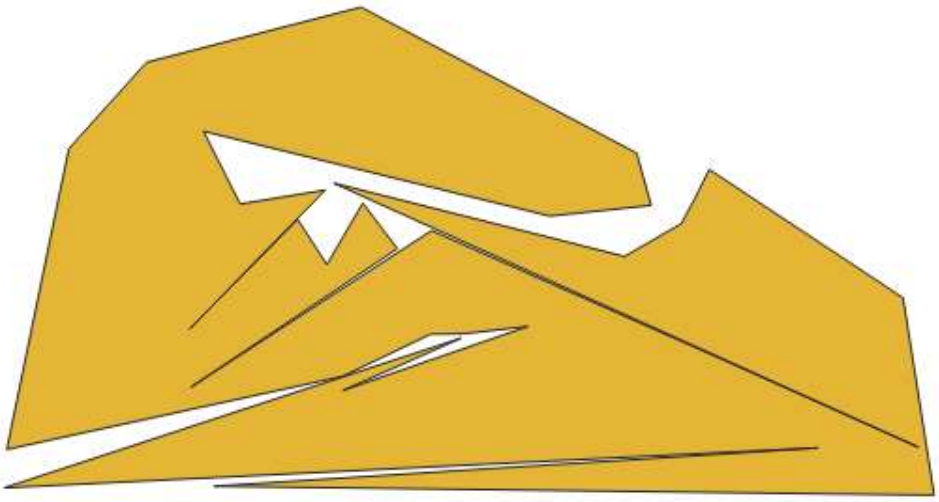
- Για τον incremental (-gen incremental) ο χρόνος εκτέλεσης είναι 15063 ms, το αρχικό ratio είναι 1.64875, το τελικό ratio είναι 1.1112 , το αρχικό εμβαδό είναι 29870856 και το τελικό εμβαδό είναι 44321170



- Για τον convex hull (-gen convex_hull) ο χρόνος εκτέλεσης είναι 4083 ms, το αρχικό ratio είναι 1.10662, το τελικό ratio είναι 1.10051 , το αρχικό εμβαδό είναι 44504504 και το τελικό εμβαδό είναι 44751756



- Για τον onion (-gen onion) ο χρόνος εκτέλεσης είναι 14361 ms, το αρχικό ratio είναι 1.65085, το τελικό ratio είναι 1.13267 , το αρχικό εμβαδό είναι 29832914 και το τελικό εμβαδό είναι 43480980



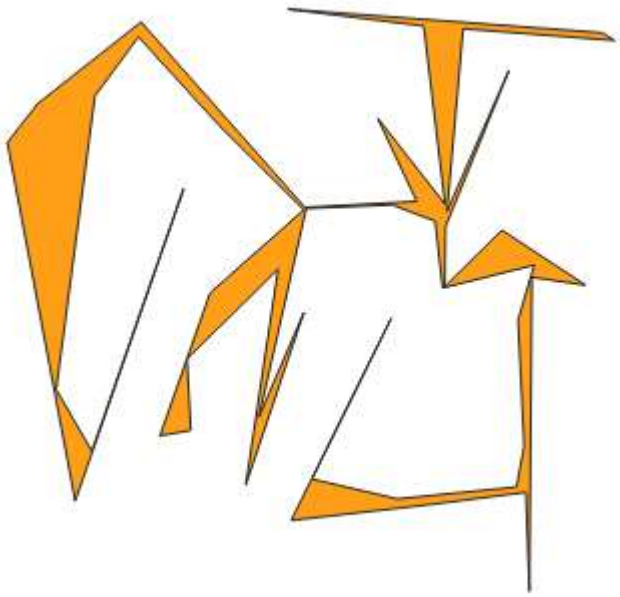
Αυτό που γίνεται κατανοητό είναι πως το κομμάτι της βελτιστοποίησης είναι πολύ πιο γρήγορο όταν το αρχικό πολύγωνο είναι «καλό». Το «καλύτερο» πολύγωνο το επιστρέφει ο convex hull με αποτέλεσμα να σπαταλάμε λιγότερο χρόνο στην βελτιστοποίηση. Επίσης, είναι πασιφανές πώς ο onion, αν και είναι ο πιο γρήγορος από τους 3 αλγορίθμους, επιστρέφει μάλλον τα «χειρότερα» πολύγωνα αφού δεν χρησιμοποιεί κάποιο άπλητο κριτήριο που να εξετάζει τα εμβαδά κατά την δημιουργία τους. Αυτό έχει ως συνέπεια και η βελτιστοποίηση που

επιστρέφεται να μην είναι η καλύτερη δυνατή συγκριτικά με αυτή που θα πετυχαίναμε αν είχαμε χρησιμοποιήσει κάποιο άλλον αλγόριθμο για το αρχικό πολύγωνο.

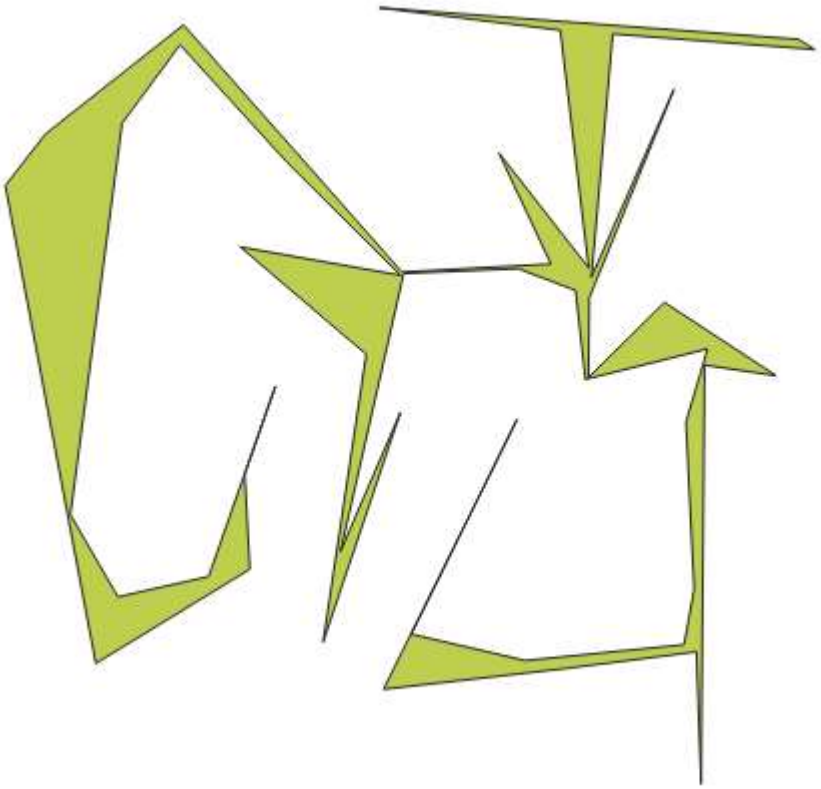
Εν συνεχεία, εξετάζουμε το μήκος της αλυσίδας (L). Αν και θα περιμέναμε μεγαλύτερο μήκος να συνεπάγεται και μεγαλύτερη βελτίωση αυτό δεν ισχύει για όλα τα σημειοσύνολα.

Για παράδειγμα για το σύνολο **stars-50** για **ελαχιστοποίηση** χρησιμοποιώντας τον convex hull για αρχικοποίηση και με threshold 0.05 έχουμε:

- Για L=1 ο χρόνος εκτέλεσης είναι 4392 ms, το αρχικό ratio είναι 3.42584, το τελικό ratio είναι 6.45362, το αρχικό εμβαδό είναι 77324937914 και το τελικό εμβαδό είναι 41047098552



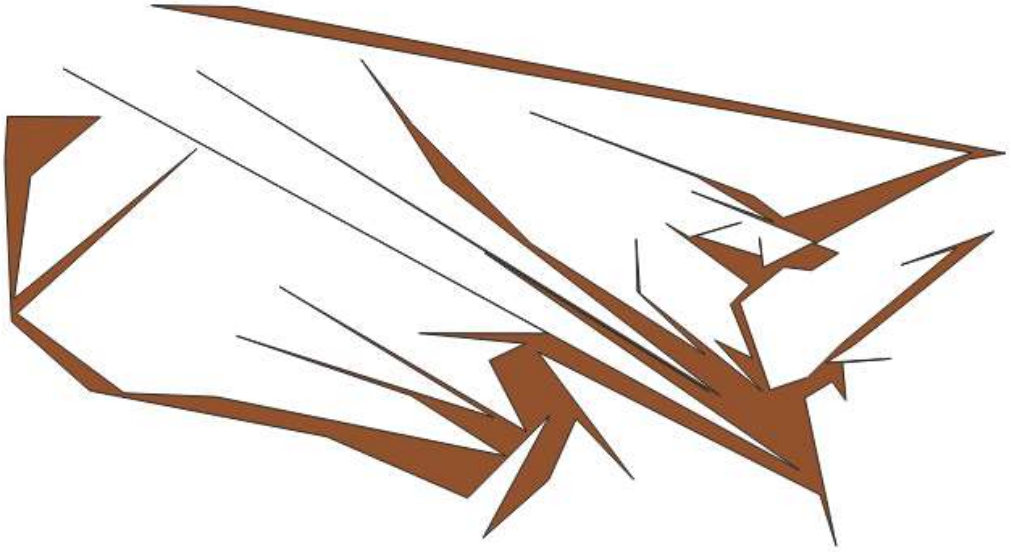
- Για L=10 ο χρόνος εκτέλεσης είναι 36822 ms, το αρχικό ratio είναι 3.42584, το τελικό ratio είναι 6.11680, το αρχικό εμβαδό είναι 77324937914 και το τελικό εμβαδό είναι 43307399664



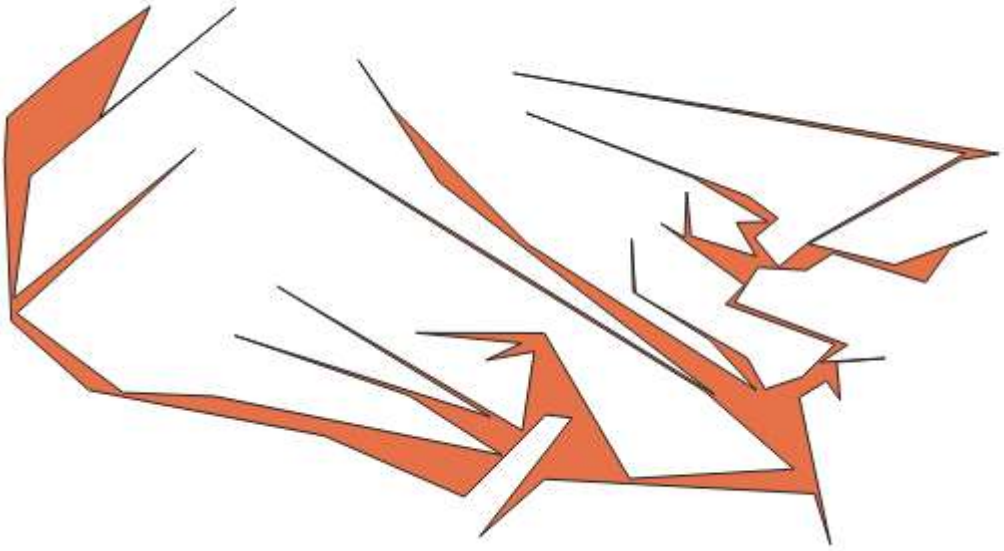
Παρόλα αυτά, για μεγαλύτερα σημειοσύνολα η συμπεριφορά είναι η αναμενόμενη.

Για παράδειγμα, για το σύνολο **us-night-90** για **ελαχιστοποίηση** χρησιμοποιώντας τον convex hull για αρχικοποίηση και με threshold 0.05 έχουμε:

- Για L=1 ο χρόνος εκτέλεσης είναι 21055 ms, το αρχικό ratio είναι 3.65987, το τελικό ratio είναι 5.62302, το αρχικό εμβαδό είναι 24712514 και το τελικό εμβαδό είναι 16084710

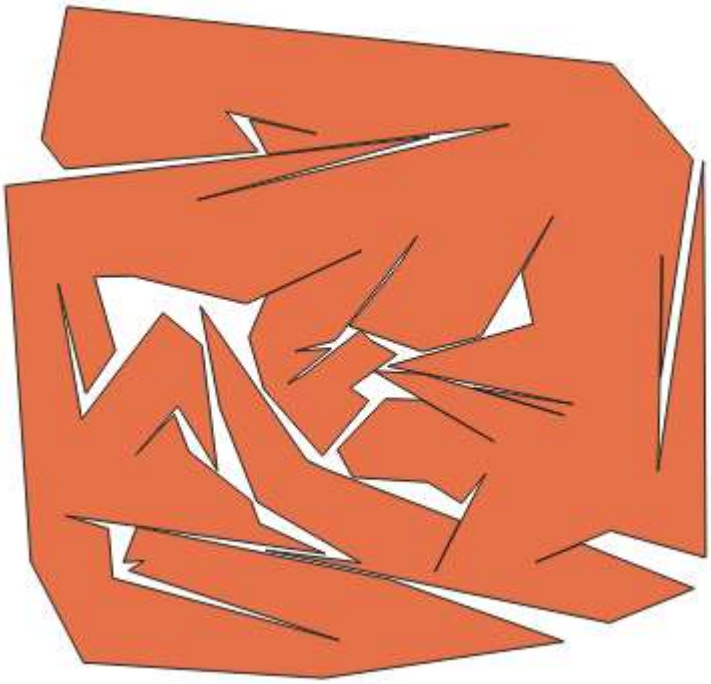


- Για L=10 ο χρόνος εκτέλεσης είναι 254863 ms, το αρχικό ratio είναι 3.65987, το τελικό ratio είναι 6.51649, το αρχικό εμβαδό είναι 24712514 και το τελικό εμβαδό είναι 13879346

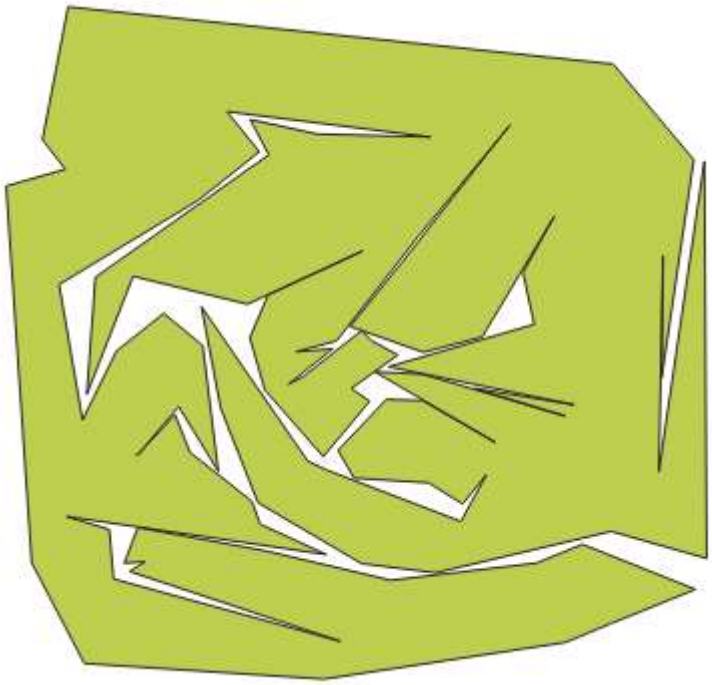


Ένα άλλο παράδειγμα είναι το αρχείο **London-100** όπου για **μεγιστοποίηση** χρησιμοποιώντας τον convex hull για αρχικοποίηση και με threshold 0.95 έχουμε:

- Για L=1 ο χρόνος εκτέλεσης είναι 30233 ms, το αρχικό ratio είναι 1.16185, το τελικό ratio είναι 1.11821, το αρχικό εμβαδό είναι 327914090 και το τελικό εμβαδό είναι 340711188



- Για L=10 ο χρόνος εκτέλεσης είναι 222339 ms, το αρχικό ratio είναι 1.16185, το τελικό ratio είναι 1.11472, το αρχικό εμβαδό είναι 327914090 και το τελικό εμβαδό είναι 341776860

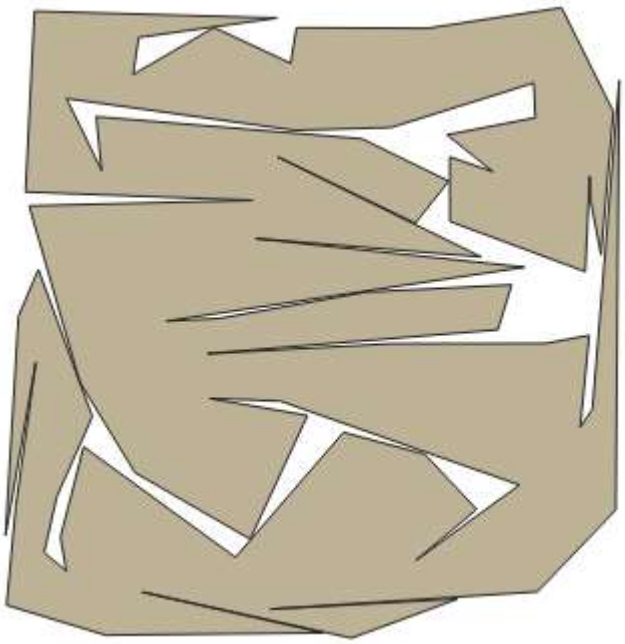


Αν υπάρχει ένα συμπέρασμα στο οποίο μπορούμε να καταλήξουμε με ασφάλεια, είναι πως η αύξηση του L οδηγεί και σε αύξηση του χρόνου εκτέλεσης, πράγμα αναμενόμενο αφού κάνουμε περισσότερες επαναλήψεις και έχουμε διαθέσιμες περισσότερες αλλαγές που πρέπει να ελέγξουμε. Αναφορικά πάντως με τα εμβαδά, φαίνεται πως η επιλογή μεγάλου L ωφελεί περισσότερο τα μεγαλύτερα σημειοσύνολα σε σχέση με τα μικρότερα.

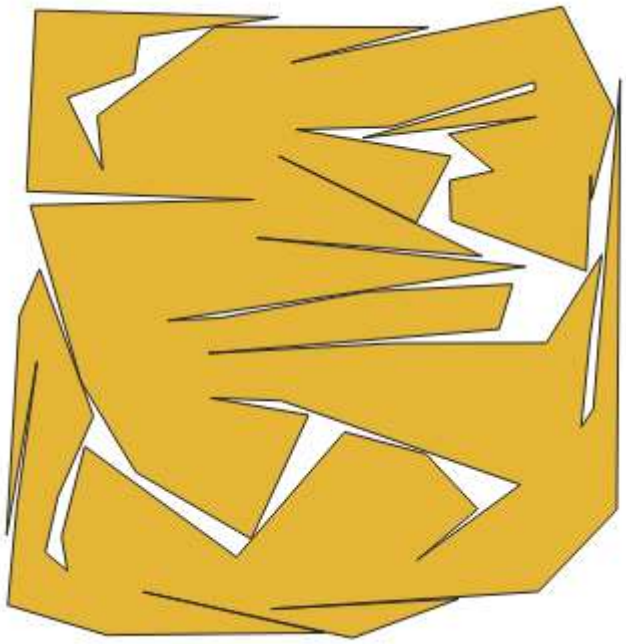
Αναφορικά με τις τιμές του threshold, η συμπεριφορά και τα αποτελέσματα είναι αυτά που αναμένονταν. «Καλύτερες» τιμές του threshold οδηγούν σε «καλύτερες» βελτιστοποιήσεις θυσιάζοντας όμως την ταχύτητα εκτέλεσης του προγράμματος αφού ο χρόνος εκτέλεσης αυξάνεται Ενδεικτικά

Για το αρχείο **uniform-80-1** όπου για **μεγιστοποίηση** χρησιμοποιώντας τον convex hull για αρχικοποίηση και με L=10 έχουμε:

- Για threshold = 0.85 ο χρόνος εκτέλεσης είναι 50461 ms, το αρχικό ratio είναι 1.20943, το τελικό ratio είναι 1.17409, το αρχικό εμβαδό είναι 17283580 και το τελικό εμβαδό είναι 17803800

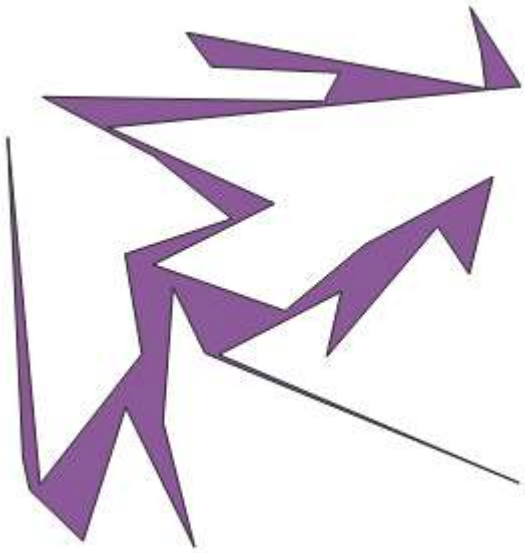


- Για threshold = 0.95 ο χρόνος εκτέλεσης είναι 138311 ms, το αρχικό ratio είναι 1.20943, το τελικό ratio είναι 1.16047, το αρχικό εμβαδό είναι 17283580 και το τελικό εμβαδό είναι 18012810

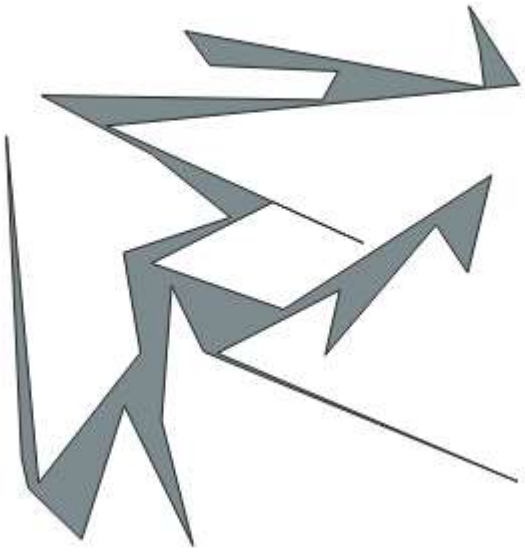


Για το αρχείο **uniform-35-1** όπου για **ελαχιστοποίηση** χρησιμοποιώντας τον convex hull για αρχικοποίηση και με L=10 έχουμε:

- Για threshold = 0.20 ο χρόνος εκτέλεσης είναι 6437 ms, το αρχικό ratio είναι 2.7631, το τελικό ratio είναι 5.12256, το αρχικό εμβαδό είναι 1255602 και το τελικό εμβαδό είναι 682958



- Για threshold = 0.05 ο χρόνος εκτέλεσης είναι 11448 ms, το αρχικό ratio είναι 2.7631, το τελικό ratio είναι 5.28980, το αρχικό εμβαδό είναι 1255602 και το τελικό εμβαδό είναι 661366



Όπως, φάνηκε και από τα παραπάνω αποτελέσματα των εκτελέσεων, ο χρόνος εκτέλεσης βασίζεται σε τουλάχιστον 3 παράγοντες, το μήκος της αλυσίδας (L), την τιμή του threshold και τον αλγόριθμο αρχικοποίησης. Προφανώς και το μέγεθος του σημειοσυνόλου επηρεάζει τον χρόνο, αλλά πρόκειται για κάτι το οποίο δεν μπορούμε να αποφύγουμε.