README

ΣΤΟΙΧΕΙΑ ΦΟΙΤΗΤΩΝ

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: ΚΥΡΙΤΣΗΣ ΧΡΗΣΤΟΣ
ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ: 1115202100075
ΟΝΟΜΑΤΕΠΩΝΥΜΟ: ΑΝΑΓΝΩΣΤΟΠΟΥΛΟΣ ΔΗΜΗΤΡΙΟΣ
ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ: 1115202100002

ΚΑΤΑΛΟΓΟΣ ΑΡΧΕΙΩΝ ΚΑΙ Ο ΡΟΛΟΣ ΤΟΥΣ

Το project στο εξωτερικό του αποτελείται από τους φακέλους triangulation, utils, input, output και build καθώς και από τα αρχεία CMakeLIsts.txt,opt_triangulation.cpp, input.json.

Στον φάκελο triangulation βρίσκονται τα αρχεία triangulation.cpp και triangulation.h τα οποία έχουν συναρτήσεις οι οποίες βοηθάνε στην επίτευξη της μη-αμβλυγώνιας τριγωνοποίησης π.χ. μετρώντας καθε φορά το πλήθος των αμβλυγώνιων τριγώνων, είτε κοιτώντας αν η εισαγωγή ενός Steiner Point θα βελτίωνε την τριγωνοποίηση κτλ. Επίσης, σε αυτόν τον φάκελο βρίσκονται και οι συναρτήσεις των τριών αλγορίθμων localSearch, simulatedAnnealing, antColonyOptimization , καθώς και όλες τις βοηθητικες συναρτήσεις που χρησιμοποιούν αυτοί οι αλγόριθμοι.

Στον φάκελο utils έχουμε τα αρχεία utils.cpp και utils.h, τα οποία ο ρόλος τους είναι να διαβάζουν και να επεξεργάζονται το input αρχείο κάθε φορά και να το τροποποιούν αναλόγως έτσι ώστε σε κάθε παράδειγμα να παίρνουμε τα κατάλληλα δεδομένα.

Στον φάκελο tests βρίσκεται η είσοδος που δίνουμε κάθε φορά στο πρόγραμμά μας.

Στον φάκελο output βρίσκεται το αποτέλεσμα της εξόδου μετά την τριγωνοποίηση με κάποιον από τους 3 αλγορίθμους.

Το αρχείο opt_triangulation.cpp έχουμε την main η οποία με την σειρά της καλεί αρχικά της συναρτήσεις που επεξεργάζονται τα αρχεία json κάνει μια αρχική

οπτικοποίηση και στην συνέχεια καλεί τη συνάρτηση για την τριγωνοποίηση.

ΟΔΗΓΙΕΣ ΜΕΤΑΓΛΩΤΙΣΗΣ ΚΑΙ ΤΡΕΞΙΜΟ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ

Για μεταγλώτιση θα πρέπει να μπούμε στον φάκελο build με την εντολή cd build και για τρέξιμο εκτελούμε την εντολή:
./opt_triangulation –i ../tests/input.json –o ../output/output.json

ΠΕΡΙΓΡΑΦΗ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ

Παρακάτω ακολουθεί η περιγραφή όλων των επιπλεόν συναρτήσεων που υλοποιήθηκαν για την δεύτερη εργασία. Οι συναρτήσεις της πρώτης περιγράφονται λεπτομερώς στο αντίστοιχο README της πρώτης.

Στο αρχείο opt_triangulation καλούνται συναρτήσεις για φόρτωση των δεδομένων από τα αρχεία json όπως η loadJsonFile και αναλύονται από την συνάρτηση parseJson της boost::json βιβλιοθήκης, η οποία τα μετατρέπει σε αντικείμενο Object. Στη συνέχεια μέσω της συνάρτησης retreiveFields ανακτώνται τα πεδία όπως instance_uid, num_points, points_x, points_y, num_constraints, additional_constraints και region_boundary. Εφόσον γίνουν οι παραπάνω διαδικασίες δημιουργούνται τα σημεία ως ζεύγη συντεταγμένων και τέλος πριν τον τερματισμό του προγράμματος καλείται η CallprocessTriangulation με σκοπό την τριγωνοποίηση.

Ο φάκελος utils έχει παραμείνει ίδιος με την πρώτη εργασία.

Στον φάκελο Triangulation προστέθηκαν πολλές συναρτήσεις για την δεύτερη εργασία, όμως οι συναρτήσεις isPointOnSegment, isPointInsideBoundary, isPointInPolygon, isTriangleWithinBoundary, squaredDistance, isObtuseTriangle, countObtuseTriangles, tryEdgeFlipping, simulateSteinerEffect, getMidpointOfLongestEdge, calculate_incenter, calculate_perpendicular_bisector_point, projectPointOntoTriangle, createOutputJson και printOutputJson εξηγούνται και περιγράφονται στο README της πρώτης εργασίας.

Η συνάρτηση EnsureConvexPolygon μέσω της CGAL::convex_hull_2 επιστρέφει τα σημεία που αποτελούν το κυρτό περίβλημα.

Η συναρτηση processConvexPolygon αναζητά αμβλείες γωνίες χωρίς constrained edges, τις αποθηκεύει και μέσω της ensureConvexPolygon εξασφαλίζει ότι οι κορυφές σχηματίζουν κυρτό πολύγωνο. Στη συνέχεια υπολογίζει το centroid του πολυγώνου, αφαιρεί τις παλιές κορυφές και εισάγει constraint edges για

να διατηρήσει το σχήμα του πολυγώνου. Τέλος προσθέτει το centroid ως νέο σημείο και το συνδέει με τις κορυφές του πολυγώνου.

Η συνάρτηση selectMethod καλεί μία από τις 3 μεθόδους τριγωνοποίησης που υλοποιήθηκαν ανάλογα με το όρισμα το οποίο έχει δωθεί και περνάει σε κάθε μία από τις συναρτήσεις και τις κατάλληλες παραμέτρους. Τέλος εξετάζει και την περίπτωση εισαγωγής λάθος μεθόδου.

Η συνάρτηση localSearch είναι η πρώτη από τις 3 μεθόδους που ζητήθηκαν, η οποία κάνει τοπική αναζήτηση για την βελτιστοποίηση της τριγωνοποίησης, ελαχιστοποιώντας τα αμβλωγώνια και βάζοντας κατάλληλα STeiner Points . Αρχικά, υπολογίζει τα αμβλυγώνια και όσο τα αμβλυγώνια δεν είναι 0 ή δεν έχουμε ξεπεράσει τον αριθμό μέγιστων επαναλήψεων που δίνεται ως όρισμα στην συνάρτηση εφαρμόζουμε τον αλγόριθμο. Πιο συγκεκριμένα εντοπίζονται τα αμβλυγωνια τρίγωνα εντός των ορίων με έγκυρες συντεταγμενες και για κάθε ένα από αυτά κοιτάμε κάθε μία από τις στρατηγικές που χρησιμοποιούμε, και επιλέγουμε το Steiner Point το οποί ο είναι έγκυρο και μειώνει τα αμβλυγώνια τρίγωνα περισσότερο από τις άλλες στρατηγικές. Τέλος, ενημερώνει κατάλληλα το σχήμα και αποθηκέυει τα κατάλληλα δεδομένα για το αρχείο εξόδου και καλεί την συνάρτηση CGAL::draw για οπτικοποίηση των αποτελεσμάτων.

Η συνάρτηση getPointIndex διατρέχει τη λίστα σημείων και συγκρίνει το x και το y του δοθέντος σημείου με τις συντεταγμένες των σημείων της λίστας. Αν δεν βρεθεί επιστρέφει -1.

Η συνάρτηση createCDT Μετατρέπει τα σημεία από τη λίστα points_ σε αντικείμενα CGAL Point και τα εισάγει στην τριγωνοποίηση. Προσθέτουμε contraint edges στην τριγωνοποίηση και επιστρέφει ολόκληρη την τριγωνοποίηση.

Η συνάρτηση simulateAnnealing εφαρμόζει την τεχνική της προσομοίωσης ανόπτησης για να μειώσει τον αριθμό των αμβλειών τριγώνων σε μια τριγωνοποίηση. Υπολογίζει αρχικά την ενέργεια και θέτει την θερμοκρασία σε 1.0. Όσο η θερμοκρασία παραμένει μεγαλύτερη ή ίση του 0 για κάθε αμβλυγώνιο τρίγωνο επιλέγει μία στρατηγική κοιτώντας αν το σημείο Steiner είναι έγκυρο το τοποθετεί στην τριγωνοποίηση και υπολογίζει την νέα ενέργεια και χρησιμοποιεί έναν έλεγχο για το αν αποδεχτούμε την νέα τριγωνοποίηση ή όχι. Έπειτα ενημερώνει την θερμοκρασία και τέλος καλέι την συνάρτηση για το αρχείο εξόδου με ορίσματα τις μεταβλήτές που αποθηκεύονται και την συνάρτηση οπτικοποίησης των αποτελεσμάτων.

Η συνάρτηση InsertSteinerPoint εισάγει ένα σημείο στην τριγωνοποίηση με την cdt.insert .

Η συνάρτηση getRandomUniform επιστρέφει ομοιόμορφα έναν τυχάιο αριθμό στο διάστημα [0, 1].

Η συνάρτηση calculateEnergy υπολογίζει την ενέργεια μιας τριγωνοποίησης η

οπόια χρησιμοποιεί τα α, β των αριθμό αμβλυγωνίων τριγώνων και από το πλήθος των σημείωνSteiner . Εδώ σε αντίθεση με τον τύπο που δίνεται στις διαφάνειες και με μια αντίστοιχη συζήτηση που είδαμε στο eclass καταλήξαμε να χρησιμοποιούμε αντί για το πλήθος αμβλυγώνιων τριγώνων να χρησιμοποιούμε το πλήθος αμβλυγώνιων τριγώνων στο τετράγωνο, καθώς έδινε καλύτερα αποτελέσματα.

Η συνάρτηση getMeanAdjacentPoint υπολογίζει το περίκεντρο του τρέχοντος τριγώνου. Στη συνέχεια για κάθε γειτονικό τρίγωνο εξετάζει αν είναι αμλυγώνιο και αν είναι υπολογίζει και αποθηκεύει το περίκεντρο του. Αν τα γειτονικά τρίγωνα είναι λιγότερα από 2 τότε επιστρέφει το σημείο (0, 0), αλλίως υπολογίζει τον μέσο όρο τον περικέντρων και το επιστρέφει αν είναι εντος ορίων.

Η συνάρτηση UpdatePheromones ενημερώνει τις φερομόνες εξατμίζωντας τις υπάρχουσες φερομόνες με συντελεστή λ. Τέλος ενισχύει τις φερομόνες ανάλογα με την βελτίωση που επέφεραν.

Η συνάρτηση applyAntSolution εισάγει το σημείο μέσω της cdt.insert .

Η συνάρτηση resolveConflicts υπάρχει για να διαχειρίζεται τυχόν συγκρούσεις σε περίπτωση που 2 ή παραπάνω μυρμήγκια επιλέξουν το ίδιο σημείο. Σε αυτή την περίπτωση απλά επιλέγει το μυρμήγκι που βελτιώνει περισσότερο την τριγωνοποίηση και ενημερώνει κατάλληλα τις επιλογές χωρίσ συγκρούσεις.

Η συνάρτηση generateSteinerOptions επιστρέφει τις επιλογές στρατηγικής για εισαγωγή σημείου Steiner.

Η συνάρτηση selectStrategy επιλέγει μια στρατηγική βάσει πιθανοτήτων από μια λίστα στρατηγικών και των αντίστοιχων πιθανοτήτων τους.

Η συνάρτηση antColonyOptimization αρχικοποιεί για κάθε τρίγωνο τις φερομόνες. Στη συνέχεια κάνει τις επαναλήψεις για κύκλους και μυρμήγκια. Κάθε κύκλος δημιουργεί τις λύσεις από κάθε μυρμήγκι, υπολογίζει τις πιθανότητες επιλογής σημείου Steiner , το οποίο βασίζεται στις φερομόνες και την ευρετική συνάρτηση και προσομοίωση της προσθήκης σημείων και καταγραφή του ενεργειακού κέρδους. Έπειτα, ενημερώνει την τρέχουσα τριγωνοποίηση με νέα αν υπάρχει βελτίωση ή η ενέργεια μειώνεται. Μετά από αυτό το βήμα ενημερώνονται οι φερομόνες ανάλογα και τέλος καλείται η συνάρτηση εξόδου η οποία περνά ως ορίσματα όλα τα απαραίτητα τα οποία αποθηκεύτηκαν μέσα στην συνάρτηση και οπτικοποιεί το αποτέλεσμα.

Η συνάρτηση copyTriangulation αντιγράφει μία τριγωνοποίηση σε μία άλλη.

Η συνάρτηση selectSteinerPoint επιστρέφει ένα σημείο με βάση τις πιθανότητες. Σε περίπτωση που δεν υπάρχουν επιλογές επιστρέφει το (0, 0)

Η συνάρτηση calculateHeuristic υπολογίζει διάφορα δεδομένα πάνω σε κάποιο

τρίγωνο και ανάλογα με την στρατηγική επιστρέφει και κάποια ευρετική τιμή

Η συνάρτηση toFraction μετατρέπει έναν δεκαδικό αριθμό σε κλάσμα.

Η συνάρτηση writeOutputToFile παίρνει τα δεδομένα, μετατρέπει τις παραμέτρους σε ένα αντικείμενο τύπου JSON , καταχωρεί τα σημεία Steiner , δημιουργεί πίνακα JSON για τις ακμές και καταχωρεί και τον αριθμών των αμβλυγώνιων τριγώνων. Στη συνέχεια γράφει τα δεδομένα στο αρχείο JSON με τον κατάλληλο τρόπο όπως ζητείται και στην εκφώνηση.

ΑΠΟΤΕΛΕΣΜΑΤΑ ΕΚΤΕΛΕΣΕΩΝ

local, delaunay = true

initial test: 21 → 11

local, delaunay = false

initial test: 21 → 10

sa, delaunay = true

initial test: 21 → 11

sa, delaunay = false

initial test: 21 → 13

ant, delaunay = true

initial test: 21 → 11

ant, delaunay = false

initial test: 21 → 17

Σε μερικά από τα υπόλοιπα test cases της μεθόδου sa παρατηρήθηκε μείωση των αμβλυγώνιων τριγώνων, κυρίως σε αυτά τα οποία η πρώτη εργασία δεν τα μείωνε καθόλου

ΚΑΠΟΙΕΣ ΕΠΙΠΛΕΟΝ ΠΑΡΑΤΗΡΗΣΕΙΣ

Σε αυτό το project παρατηρήσαμε πως ο αριθμός των επαναλήψεων δεν καθορίζει απαραίτητα τα αποτελέσματα. Δηλαδή, δεν είναι πάντα η καλύτερη πρακτική να αυξήσουμε όσο το δυνατόν περισσότερο τις επαναλήψεις για να δούμε βέλτιστα αποτελέσματα αλλά ίδια αποτελέσματα μπορούν να επιτευχθούν με μικρότερο αριθμό επαναλήψεων

Μια ακόμη παρατήρηση είναι πως για ίδιο input μπορούμε να πάρουμε διαφορετικό output . Αυτο συμβαίνει πχ στην ant όπου ο αριθμός των αμβλυγωνλιων τριγώνων διαφέρει απο εκτέλεση σε εκτέλεση με την ίδια ακριβώς είσοδο.