Ανάπτυξη Λογισμικού για Δύσκολα Αλγοριθμικά Προβλήματα 3 2024-25

Table of Contents

1. Περιγραφή του προγράμματος

Το πρόγραμμα υλοποιεί μια διαδικασία τριγωνοποίησης με στόχο την προσθήκη σημείων Steiner για τη μετατροπή αμβλύγωνων τριγώνων σε μη αμβλύγωνα. Χρησιμοποιεί τη βιβλιοθήκη CGAL για την υλοποίηση της τριγωνοποίησης Delaunay και την επεξεργασία γεωμετρικών σχημάτων.

Λειτουργίες του προγράμματος της πρώτης εργασίας

1. **Τριγωνοποίηση Delaunay**: Χρησιμοποιεί την τριγωνοποίηση Delaunay με περιορισμούς για τη δημιουργία ενός πλέγματος από τρίγωνα με βάση τα σημεία εισόδου.
2. **Προσθήκη σημείων Steiner**: Υπάρχουν διαφορετικές μέθοδοι για την μείωση αμβλύων γωνιών όπως περιστροφή ακμών και διάφορες τεχνικές τοποθέτησης σημείων Steiner, οι οποίες περιγράφονται σε διαφορετικά αρχεία.
3. **Εξερεύνηση και επεξεργασία τριγώνων**: Το πρόγραμμα τρέχει επαναληπτικά για όλα τα αμβλυγώνια τρίγωνα του σχήματος. Ελέγχει αν ένα τρίγωνο είναι αμβλύ, και προσθέτει σημεία Steiner μέχρι να μην υπάρχουν αμβλύγωνα τρίγωνα στο πλέγμα.
4. **Ανάγνωση και Εγγραφή Δεδομένων**: Διαχειρίζεται αρχεία JSON για την είσοδο και την έξοδο των δεδομένων, επιτρέποντας την εισαγωγή γεωμετρικών σχημάτων και την εξαγωγή των αποτελεσμάτων.
5. **Σχεδίαση και Οπτικοποίηση**: Χρησιμοποιεί τη δυνατότητα σχεδίασης της CGAL για την οπτικοποίηση των τριγωνοποιημένων σχημάτων, επιτρέποντας στον χρήστη να δει την αρχική κατάσταση του σχήματος και την τελική που δημιουργείται από την προσθήκη σημείων Steiner.
6. **Επιπλέον χαρακτηριστικά του προγράμματος που αξίζει να σημειωθούν**:
    i. Έχουμε ορίσει τον αριθμό επαναλήψεων για τα Steiner σημεία σε 5, έτσι ώστε να μην μπει το πρόγραμμα σε infinite loop σε περίπτωση που η εισαγωγή Steiner σημείων δεν καταλήξει ποτέ σε σχήμα χωρίς αμβλείες γωνίες.
    ii. Παρατηρήσαμε ότι η συνάρτηση flip_edges δεν καταλήγει ποτέ σε βελτίωση των αμβλείων γωνιών.
    iii. Η προβολή φαίνεται να λειτουργεί γενικά σαν μέθοδος, αφού αν λυθεί εσωτερικά το θέμα των αμβλείων γωνιών, στο κυρτό περίβλημα η τεχνική αυτή απαλείφει σίγουρα τις αμβλείες γωνίες διότι το σημείο δημιουργεί απλά δυο ορθογώνια και σταματάει να επεκτείνεται.

Έξτρα λειτουργίες

1. **Τοπική Αναζήτηση:** Επαναληπτική εισαγωγή σημείων Steiner σε 5 πιθανές θέσεις, επιλέγοντας τη βέλτιστη για μείωση αμβλυγώνιων.
2. **Προσομοιωμένη Ανόπτηση:** Υπολογισμός «ενέργειας» τριγωνοποίησης (alpha: αμβλυγώνια, beta: Steiner). Εξετάζονται 5 θέσεις, ενώ χειρότερες επιλογές μπορεί να γίνουν αποδεκτές βάσει του κριτηρίου **Metropolis**.

3. **Βελτιστοποίηση με Αποικία Μυρμηγκιών**
   Οι πράκτορες (μυρμήγκια) εντοπίζουν θέσεις για εισαγωγή Steiner βάσει ευρετικής και ίχνους φερομόνης.

4. **Επιπλέον χαρακτηριστικά του προγράμματος που αξίζει να σημειωθούν**:

   - Το πρόγραμμα έχει καλύτερα αποτελέσματα στην περίπτωση που ξεκινάμε από delaunay τριγωνοποίηση.
   - Στο αρχείο output.json παρουσιάζουμε και μια μεταβλητή (energy) για το ρυθμό σύγκλισης.
   - Έχουμε παρατηρήσει ότι τα σχήματα έχουν καλύτερα αποτελέσματα για ~10 επαναλήψεις (μεταβλητή L). Στο αρχείο των αποτελεσμάτων μας έχουμε χρησιμοποιήσει την τιμή L=30 (στις μεθόδους ant colony & simulated annealing) για να δοκιμάσουμε το μέγιστο δυνατό αριθμό επαναλήψεων (30). Για αριθμούς μεγαλύτερους από αυτόν, το πρόγραμμα καθυστερεί κατά την εκτέλεση.

2. Κατάλογο των αρχείων κώδικα και το ρόλο τους

Αρχεία πρώτης εργασίας

- **center.cpp**
  Στο συγκεκριμένο αρχείο η τοποθέτηση των Steiner σημείων στα αμβλυγώνια τρίγωνα γίνεται επιλέγοντας για κάθε τέτοιο τρίγωνο το μέσο της πλευράς που βρίσκεται απέναντι από την αμβλεία γωνία.
- **center.h**
  Ορισμός center_steiner_points συνάρτησης.
- **centroid.cpp**
  Στο συγκεκριμένο αρχείο η τοποθέτηση των Steiner σημείων στα αμβλυγώνια τρίγωνα γίνεται επιλέγοντας για κάθε τέτοιο τρίγωνο το βαρύκεντρό του.
- **centroid.h**
  Ορισμός centroid_steiner_points συνάρτησης.
- **circumcenter.cpp**
  Στο συγκεκριμένο αρχείο η τοποθέτηση των Steiner σημείων στα αμβλυγώνια τρίγωνα γίνεται επιλέγοντας για κάθε τέτοιο τρίγωνο το περίκεντρό του.
- **circumcenter.h**
  Ορισμός circumcenter_steiner_points συνάρτησης.
- **CMakeLists.txt**
  Αυτό το αρχείο κατασκευάζει ένα πρόγραμμα που χρησιμοποιεί τις βιβλιοθήκες CGAL, Boost, Qt5, OpenGL και GLEW. Ορίζει την ελάχιστη απαιτούμενη έκδοση CMake, ελέγχει αν οι απαιτούμενες βιβλιοθήκες είναι διαθέσιμες, και δημιουργεί έναν εκτελέσιμο αρχείο με τα κατάλληλα πηγαία αρχεία.
- **flipEdges.cpp**
  Στο συγκεκριμένο αρχείο γίνεται περιστροφή κάθε ακμής που ορίζουν δύο αμβλυγώνια τρίγωνα προκειμένου να μειωθούν συνολικά οι αμβλείες γωνίες του σχήματος.
- **flipEdges.h**

Ορισμός flip_edges συνάρτησης.

- **input.json**
  Περιέχει σε συγκεκριμένη μορφή το σχήμα εισόδου.
- **inputs.cpp**
  Συνάρτηση για να παρθούν τα πεδία του input.json και να τα χρησιμοποιήσουμε στον κώδικά μας. Αυτό γίνεται με τη συνάρτηση inputs.
- **inputs.h**
  Ορισμός inputs συνάρτησης.
- **inside_convex_polygon_centroid.cpp**
  Στο συγκεκριμένο αρχείο η τοποθέτηση των Steiner σημείων στα αμβλυγώνια τρίγωνα γίνεται επιλέγοντας για κάθε κυρτό πολύγωνο που σχηματίζεται από γειτονικά αμβλυγώνια τρίγωνα, το κέντρο του.
- **inside_convex_polygon_centroid.h**
  Ορισμός inside_convex_polygon_centroid_steiner_points συνάρτησης.
- **main.cpp**
  Το συγκεκριμένο αρχείο διαβάζει δεδομένα από αρχείο input.json, κατασκευάζει μια περιορισμένη Delaunay Τριγωνοποίηση (CDT) και εισάγει σημεία Steiner για τη βελτίωση της τριγωνοποίησης. Περιέχει επιλογές μεθόδων εισαγωγής Steiner και αλγορίθμους βελτιστοποίησης (τοπική αναζήτηση, προσομοιωμένη ανόπτηση, αποικία μυρμηγκιών). Τα αποτελέσματα αποθηκεύονται στο αρχείο output.json.
- **output.cpp**
  Συνάρτηση για να εμφανιστούν τα αποτελέσματα του κώδικά μας όπως ζητείται στο αρχείο output.json. Αυτό γίνεται με τη συνάρτηση output.
- **output.h**
  Ορισμός output συνάρτησης.
- **output.json**
  Περιέχει σε συγκεκριμένη μορφή το σχήμα εξόδου.
- **projection.cpp**
  Στο συγκεκριμένο αρχείο η τοποθέτηση των Steiner σημείων στα αμβλυγώνια τρίγωνα γίνεται επιλέγοντας για κάθε τέτοιο τρίγωνο την προβολή της αμβλείας γωνίας του. Η συνάρτηση projection που κάνει αυτή τη διαδικασία καλείται από τη main.cpp.
- **projection.h**
  Ορισμός projection συνάρτησης.

Έξτρα αρχεία δεύτερης εργασίας

- **ant_colony.cpp**
  Ο βασικός στόχος του ant_colony.cpp είναι η βελτιστοποίηση της τριγωνοποίησης μέσω της προσθήκης **Σημείων Steiner**, καθοδηγούμενη από ευρετικές και πιθανοτικές μεθόδους.

  Συναρτήσεις που χρησιμοποιήθηκαν

  - **height**: Υπολογίζει το ύψος ενός σημείου σε σχέση με την μεγαλύτερη πλευρά του τριγώνου.

- **radius_to_height_ratio**: Υπολογίζει τον λόγο ακτίνας περιγεγραμμένου κύκλου προς το ύψος του τριγώνου.
- **calculateDelta**: Υπολογίζει ενημερώσεις φερομόνης βασισμένες στις αλλαγές ποιότητας της τριγωνοποίησης.
- **steiner_point_probability**: Υπολογίζει πιθανότητες επιλογής για κάθε μέθοδο τοποθέτησης σημείων Steiner.
- **steiner_method**: Επιλέγει μέθοδο τοποθέτησης πιθανοτικά.
- **calculateEnergyAnt**: Υπολογίζει την "ενέργεια" της τρέχουσας τριγωνοποίησης.

Λειτουργικότητα

Εκτελείται μια επανάληψη για κάθε cycle:

- Για κάθε "μυρμήγκι":
  - Εντοπίζει ένα αμβλυγώνιο τρίγωνο το οποίο αντιστοιχεί στο εκάστωτε μυρμήγκι
  - Επιλέγει πιθανοτικά μια μέθοδο τοποθέτησης σημείου Steiner.
  - Εισάγει το σημείο Steiner και αξιολογεί την ενέργεια της τριγωνοποίησης.
  - Διατηρεί την αλλαγή αν βελτιώνει την ποιότητα της τριγωνοποίησης.
- Ενημερώνει τις φερομόνες με βάση τις επιτυχίες κάθε μεθόδου.

- **ant_colony.h**
  Ορισμός των συναρτήσεων που χρησιμοποιούνται στο ant_colony.cpp.
- **local_search.cpp**
  Συναρτήσεις που χρησιμοποιήθηκαν
  - **add_best_steiner:** Ανάμεσα σε 5 μεθόδους προσθήκης steiner (προβολή, περίκεντρο, βαρύκεντρο, μέσο μεγαλύτερης πλευράς, εσωτερικό πολυγώνου από γειτονικά τρίγωνα) σημείων επιλέγει την καλύτερη, δηλαδή αυτή που μειώνει περισσότερο τα αμβλεία τρίγωνα.

  Λειτουργικότητα

  Για κάθε αμβλυγώνιο τρίγωνο του σχήματος καλούμε την add_best_steiner και μας επιστρέφει την καλύτερη μέθοδο steiner για το συγκεκριμένο τρίγωνο. Αυτό γίνεται για όσο έχουμε αμβλυγώνια τρίγωνα, και αν βλέπουμε ότι τα αμβλυγώνια αυξάνονται παραπάνω από 2 φορές σταματάει η διαδικασία.

- **local_search.h**
  Ορισμός των συναρτήσεων που χρησιμοποιούνται στο local_search.cpp.
- **simulated_annealing.cpp**
  Συναρτήσεις που χρησιμοποιήθηκαν

- **calculateEnergy:** Υπολογίζει την ενέργεια βάσει των οξέων τριγώνων και του πλήθους σημείων Steiner με συντελεστές α και β.
- **generate_random_number:** Δημιουργεί έναν τυχαίο αριθμό μεταξύ 1 και 5 για την επιλογή τεχνικής δημιουργίας σημείων Steiner.
- **accept_new_configuration:** Αποφασίζει αν θα γίνει αποδεκτή μια νέα κατάσταση βάσει της μείωσης ενέργειας και της πιθανότητας μέσω της θερμοκρασίας (T).

Λειτουργικότητα

-Για κάθε αμβλυγώνιο τρίγωνο:

- Επιλέγεται πιθανοτικά μία από τις πέντε μεθόδους για την τοποθέτηση ενός σημείου Steiner:
  1. Προβολή σημείου πάνω στην πλευρά.
  2. Περίκεντρο.
  3. Βαρύκεντρο.
  4. Μέσο μακρύτερης πλευράς.
  5. Κέντρο βάρους κυρτού πολυγώνου από γειτονικά τρίγωνα.
- Εισάγεται το σημείο Steiner στην τριγωνοποίηση και υπολογίζεται η νέα ενέργεια.
- Διατηρείται η αλλαγή αν η νέα ενέργεια είναι καλύτερη ή αν γίνεται αποδεκτή πιθανοτικά (βάσει θερμοκρασίας και διαφοράς ενέργειας).

-Ενημέρωση θερμοκρασίας: Η θερμοκρασία μειώνεται σε κάθε επανάληψη.

- **simulated_annealing.h**
  Ορισμός των συναρτήσεων που χρησιμοποιούνται στο simulated_annealing.cpp.
- **utils.cpp**
  Αρχείο που περιέχει συνάρτήσεις που χρησιμοποιούνται σε διάφορα σημεία του προγράμματος.
- **utils.h**
  Ορισμός των συναρτήσεων που χρησιμοποιούνται στο utils.cpp.

3. Οδηγίες μεταγλώττισης

```
cd executables
mkdir build
cd build
cmake ..
make
```

4. Οδηγίες χρήσης του προγράμματος

Εντολή για να εκτελεστεί το πρόγραμμα: ./opt_triangulation -i ../input.json -o ../output.json

Αρχικά εμφανίζεται το μήνυμα “Category:” ακολουθούμενο από A ή B ή C ή D ή E ανάλογα με την κατηγορία στην οποία ανήκει (βάση της εκφώνησης) το σχήμα στο **input.json** αρχείο.

Στη συνέχεια αν η τιμή της μεταβλητής **delaunay** στο **input.json** είναι:

A. **false** (ο αλγόριθμος ξεκινάει από την τριγωνοποίηση της άσκησης 1) εμφανίζεται στην γραμμή εντολών το παρακάτω μήνυμα:

Please choose a method for Steiner points or Flip Edges from the following options:

1: Center of longest edge
2: Projection
3: Circumcenter
4: Centroid of internal convex polygon
5: Centroid
6: Flip Edges
Enter the number corresponding to your choice: *<waits for your choice>*

Αφού επιλέξεις μέθοδο ("method": "local" για local search, "sa" για simulated annealing, "ant" για ant colony), σου εμφανίζεται στην οθόνη ένα αρχικό σχήμα και στη συνέχεια πατώντας χ στη γραφική του παράσταση σου εμφανίζεται μια δεύτερη που παρουσιάζει το ίδιο σχήμα μετά την ενέργεια που διάλεξες. Ξαναπατώντας το χ σου εμφανίζει ένα τρίτο σχήμα με τριγωνοποίση μετά την εφαρμογή του αλγόριθμου καθολικής βελτιστοποίησης που επιλέχτηκε στο **input.json**. Στο **output.json** εμφανίζεται το output με τη μορφή που ζητήθηκε στην εκφώνηση μετά την εφαρμογή του αλγόριθμου καθολικής βελτιστοποίησης που επιλέχτηκε στο **input.json.**

B. **true** (ο αλγόριθμος ξεκινάει από delauny τριγωνοποίηση)

Σου εμφανίζεται στην οθόνη ένα αρχικό σχήμα και στη συνέχεια πατώντας χ στη γραφική του παράσταση σου εμφανίζεται μια δεύτερη με τριγωνοποίση μετά την εφαρμογή του αλγόριθμου καθολικής βελτιστοποίησης που επιλέχτηκε στο **input.json**. Στο **output.json** εμφανίζεται το output με τη μορφή που ζητήθηκε στην εκφώνηση μετά την εφαρμογή του αλγόριθμου καθολικής βελτιστοποίησης που επιλέχτηκε στο **input.json.**

5. Σχολιασμός αποτελεσμάτων

Κατηγορία Α: Για τα σχήματα της κατηγορίας αυτής (βλ. Σχήμα A results.xlsx) φαίνεται να λειτουργεί καλύτερα η μέθοδος ant colony και συγκεκριμένα όταν αυτή ξεκινάει από την delaunay τριγωνοποίηση.

Κατηγορία Β: Για τα σχήματα της κατηγορίας αυτής (βλ. Σχήμα B results.xlsx) παρατηρήσαμε τα καλύτερα αποτελέσματα με τη χρήση της ant colony και συγκεκριμένα όταν αυτή ξεκινάει από την delaunay τριγωνοποίηση ή τη μη delaunay με εισαγωγή steiner point ως προβολή. Επίσης παρατηρήσαμε καλά αποτελέσματα και με τη χρήση της local search όταν αυτή ξεκινάει από την delaunay τριγωνοποίηση ή τη μη delaunay με εισαγωγή steiner point ως προβολή.

Κατηγορία Γ: Για τα σχήματα της κατηγορίας αυτής (βλ. Σχήμα C results.xlsx) την καλύτερη επίδοση είχε η μέθοδος ant colony όταν ξεκινάει από την delaunay τριγωνοποίηση ενώ λίγο χειρότερη είχε η ίδια μέθοδος όταν ξεκινάει από τη μη delaunay τριγωνοποίηση με εισαγωγή steiner point ως

προβολή. Καλή επίδοση έχει και η local search όταν ξεκινάει από την delaunay τριγωνοποίηση ή τη μη delaunay με εισαγωγή steiner point ως προβολή.

Κατηγορία Δ: Για τα σχήματα της κατηγορίας αυτής (βλ. Σχήμα D results.xlsx) παίρνουμε τα καλύτερα αποτελέσματα όταν χρησιμοποιούμε την ant colony και συγκεκριμένα όταν αυτή ξεκινάει από την delaunay τριγωνοποίηση ή τη μη delaunay με εισαγωγή steiner point ως προβολή. Επιπλέον έχουμε αρκετά καλά αποτελέσματα με τις μεθόδους ant colony και local search όταν ξεκινάνε από όλες τις περιπτώσεις steiner point εκτός του βαρυκέντρου.

Κατηγορία Ε: Για τα σχήματα της κατηγορίας αυτής (βλ. Σχήμα E results.xlsx) παίρνουμε τα καλύτερα αποτελέσματα όταν χρησιμοποιούμε την ant colony και συγκεκριμένα όταν αυτή ξεκινάει από τη μη delaunay με εισαγωγή steiner point ως προβολή. Επιπροσθέτως έχουμε αρκετά καλά αποτελέσματα με τις μεθόδους ant colony και local search όταν ξεκινάνε από όλες τις περιπτώσεις steiner point εκτός του κέντρου της μεγαλύτερης πλευράς και του βαρυκέντρου.

6. Πλήρη στοιχεία των φοιτητών που ανέπτυξαν την εργασία

- Αντώνιος Κουρκουλάκος 1115201900239
- Μαρία Μιχαηλίδη 1115201900115

Η υλοποίηση της εργασίας έγινε μέσω GitHub και χρησιμοποιήθηκε το παρακάτω repository: AntKourk/Project3_2024-2025.