2017

Συσταδοποίηση οδικών τμημάτων

Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα
Μέρος 3ο

Κανελλόπουλος Στέφανος

(Α.Μ. 1115201200050)

Χανιωτάκης-Ψύχος Χαρίδημος

(Α.Μ. 1115201200194)

Περιγραφή προγράμματος

Αρχικά, το πρόγραμμα μας διαβάζει τα δεδομένα από το input αρχείο και τα αποθηκεύει σε έναν πίνακα μεγέθους όσο το πλήθος των segments. Στην συνέχεια κατασκευάζουμε έναν πίνακα αποστάσεων, ο οποίος περιλαμβάνει τις αποστάσεις για κάθε segment από όλα τα υπόλοιπα segments. Με αυτό τον τρόπο υπολογίζουμε μόνο μια φορά τις αποστάσεις μειώνοντας τον χρόνο εκτέλεσης του προγράμματος.

Το επόμενο βήμα είναι η εκτέλεση της διαδικασίας του clustering τύπου k-medoids με τους πιο αποτελεσματικούς αλγορίθμους, οι οποίοι είχαν υλοποιηθεί στο προηγούμενο κομμάτι του project. Δοκιμάζουμε clustering με διαφορετικό αριθμό clusters, ξεκινώντας από το ¼ του συνόλου των segments και σε κάθε βήμα κάνουμε clustering για το μισό πλήθος clusters από το προηγούμενο. Αυτό συνεχίζεται για clustering με αριθμό clusters μεγαλύτερο του δυο.

Η διαδικασία του clustering τερματίζεται υπό την συνθήκη ότι τα κέντρα είτε δεν μετατοπίστηκαν καθόλου είτε μετατοπίστηκαν ελάχιστα (συμβολική σταθερά THRESHOLD). Αφού τερματιστεί η διαδικασία, υπολογίζεται το silhouette για το σύνολο των clusters και το clustering με την καλύτερη τιμή silhouette αποθηκεύεται σε ένα αρχείο με όνομα ‘kmeans_ways_frechet.dat’ ή ‘kmeans_ways_dtw.dat’.

Δημιουργία dataset

1. Απο το αρχειο ‘athens_greece.osm’ δημιουργουμε το αρχειο ‘athens_street.osm’ που περιλαμβανει μονο τα highways των ζητουμενων τυπων. Αυτο επιτυγχανεται με την παρακατω εντολη:

   « *osmfilter athens_greece.osm --keep-nodes-ways="highway=service =motorway =trunk =primary =secondary =tertiary =unclassified =residential" > athens_streets.osm* »

2. Με την χρηση της Python και της βιβλιοθηκης pandas απο το αρχειο ‘athens_streets.osm’ δημουργουμε το ‘athens.csv’ που περιλαμβανει ολα τα στοιχεια που ζητηθηκαν (way id, way type, lat/lon coordinates). Ο κωδικας που χρησιμοποιηθηκε υπαρχει διαθεσιμος.

3. Για το δευτερο σκελος απο το αρχειο ‘athens_streets.osm’ παλι με τη χρηση python δημιουργησαμε το segments.csv το οποιο συμπεριλαμβανει τα segments στα οποια χωριστηκαν οι δρομοι .

Πιο συγκριμενα, αρχικα μετρησαμε τις εμφανισεις του καθε node ετσι ωστε να γνωριζουμε ποιο αποτελει διασταυρωση. Με τη χρηση της python χωρισαμε τους δρομους πρωτα με βαση τις διασταυρωσεις , παιρνοτας υποψη μας αν υπαρχουν διαδοχικες διασταυρωσεις , η κατατμηση γινεται στη τελευταια διασταυρωση. Φροντιζουμε να μην υπαρχουν segments με λιγοτερα απο 2 Nodes.
Στο δευτερο σκελος της επεξεργασιας τα segments χωριζονται με βαση τη καμπυλοτητα της καμπυλης της οποιας αποτελουν (καθε segment αποτελει καμπυλη). Οταν μια καμπυλη αλλαζει αποτομα καμπυλοτητα συμφωνα με τις οδηγιες που δοθηκαν θα πρεπει να κατατμειται σαυτο το σημειο. Αυτο το σημειο κατατμησης ειναι ουσιαστικα ενα threshold το οποιο ορισαμε βρισκοντας τη μεση καμπυλοτητα των καμπυλων ολου του dataset

Αρχεία κώδικα και σύντομη περιγραφή τους

- makefile: Μεταγλώττιση προγράμματος και παραγωγή εκτελέσιμου αρχείου
- main.c: Δέχεται τα ορίσματα του χρήστη, δημιουργεί τον πίνακα με τα δεδομένα και καλεί τις συναρτήσεις για clustering για διαφορετικό αριθμό clusters κάθε φορά
- structs.c: Περιλαμβάνει όλες τις δομές που ήταν απαραίτητες για την υλοποίηση του προγράμματος
- functions.c: Περιλαμβάνει βοηθητικες συναρτήσεις που είναι απαραίτητες για την εκτελεση του clustering
- functions.h: Περιλαμβάνει όλες τις standard libraries, όπως και τις συμβολικές σταθερές
- cRMSD.c: Περιλαμβάνει την υλοποιηση της συναρτησης c-RMSD με την χρήση των βιβλιοθηκών lapacke και cblas
- cRMSD.h: Αρχείο επικεφαλίδας του cRMSD.c
- metric_functions.c: Περιλαμβάνει τις συναρτήσεις για τον υπολογισμό της απόστασης frechet μεταξύ δύο conformations.
- metric_functions.h: Αρχείο επικεφαλίδας του metric_functions.c
- input_functions.c: Περιλαμβανει τις συναρτησεις για το διαβασμα των δεδομενων από το input file
- input_function.h: Αρχείο επικεφαλίδας του input_functions.c
- output_functions.c: Περιλαμβάνει την συναρτήση για την εξαγωγή των αποτελεσμάτων στο output file
- output_functions.h: Αρχείο επικεφαλίδας του output_functions.c
- quicksort.c: Περιλαμβάνει τις συναρτήσεις για την υλοποίηση της quicksort για ταξινόμηση μονοδιάστατου πίνακα από integers
- quicksort.h: Αρχείο επικεφαλίδας του quicksort.c
- clustering_init.c: Περιλαμβάνει την υλοποίηση της συνάρτησης k-means++
- clustering_init.h: Αρχείο επικεφαλίδας του clustering_init.c
- clustering_assignment.c: Περιλαμβάνει την υλοποίηση της συνάρτησης lloyd
- clustering_assignment.h: Αρχείο επικεφαλίδας του clustering_assignment.c
- clustering_update.c: Περιλαμβάνει την υλοποίηση της συνάρτησης PAM
- clustering_update.h: Αρχείο επικεφαλίδας του clustering_update.c
- silhouette.c: Περιλαμβάνει την υλοποίηση της μετρικής αποδοτικότητας του clustering
- silhouette.h: Αρχείο επικεφαλίδας του silhouette.c

Μεταγλώττιση και χρήση του προγράμματος

Για την μεταγλώττιση του προγράμματος περιλαμβάνεται αρχείο makefile και πραγματοποιείται με την εντολή "./make" στον τρέχοντα κατάλογο. Με την εντολή "./make clean" γίνεται ο καθαρισμός των αρχείων αντικειμένων.

Το πρόγραμμα εκτελείται με την παρακάτω εντολή:

```
./roads –i <input file> –function {Frechet, DTW}
```

Σε περίπτωση που κάποιο από τα παραπάνω ορίσματα δεν δοθεί στο command line, ο χρήστης έχει την δυνατότητα σε runtime να τα δώσει (θα ζητηθούν όλα όσα λείπουν).

Valgrind – Memory leaks

Έγινε έλεγχος με valgrind για memory leaks και τα αποτελέσματα φαίνονται παρακάτω.

- Για την μετρικη συναρτηση Dynamic Time Warping (DTW) εχουμε:

```
==4032==
==4032== HEAP SUMMARY:
==4032==     in use at exit: 11,912,948 bytes in 354,845 blocks
==4032==   total heap usage: 35,094,617 allocs, 34,739,772 frees, 2,214,554,292 bytes allocated
==4032==
==4032== LEAK SUMMARY:
==4032==    definitely lost: 11,753,728 bytes in 349,066 blocks
==4032==    indirectly lost: 159,092 bytes in 5,776 blocks
==4032==      possibly lost: 128 bytes in 3 blocks
==4032==    still reachable: 0 bytes in 0 blocks
==4032==         suppressed: 0 bytes in 0 blocks
==4032== Rerun with --leak-check=full to see details of leaked memory
```

Παρατηρούμε ότι το συγκεντρωτικό memory leak του προγράμματος ανέρχεται στα 11.912.948 bytes από τα συνολικά 2.214.554.292 bytes, που αντιστοιχούν στο 0,005% της συνολικής μνήμης που χρησιμοποιήθηκε.