Το αρχείο ***REPORT*** περιέχει αναλυτικές πληροφορίες και για τις δύο πρώτες εργασίες.

Εδώ θα βρείτε πληροφορίες για την 3η εργασία

Από την προηγούμενη εργασία προσθέσαμε το αρχείο ***evaluate.cpp*** και τα ***includes/evaluate_helpers/evaluate.hpp\cpp.*** Το αρχείο CMakeLists έχει τροποποιηθεί κατάλληλα ώστε να δημιουργείται και το εκτελέσιμο ***evaluate*** με την κλήση του make. Επίσης συμπεριλαμβάνουμε το αρχείο eval.out με ενδεικτικά αποτελέσματα εκτέλεσης σε αρχεία εισόδου μεγέθους από 10 έως 700.

Στο πρώτο περιέχεται η κυρίως εφαρμογή . Εκτελεί τους αλγορίθμους που θεωρούμε ότι δίνουν τα καλύτερα αποτελέσματα στα αρχεία ενός φακέλου, που δίνεται ως είσοδος, και σώζει τα αποτελέσματα των εκτελέσεων σε ένα αρχείο, που δίνεται επίσης ως είσοδος, με την μορφοποίηση που ορίζεται στην εκφώνηση.

Στα δεύτερα αρχεία περιέχονται βοηθητικές συναρτήσεις για την υλοποίηση της κυρίως εφαρμογής και έχει οργανωτική λειτουργία.

Συγκριτική αξιολόγηση και βελτιώσεις:

Σχόλια για την αξιολόγηση και σύγκριση των αλγορίθμων θα βρείτε στο ***REPORT***. Στην 2η εργασία πραγματοποιήσαμε μια εκτενή ανάλυση όλων των αλγορίθμων με περισσότερα κριτήρια και παραμέτρους από αυτές που παράγει η εφαρμογη της εργασίας 3. Θεωρήσαμε ότι δεν υπάρχει κάποια επιπλέον ανάλυση να γίνει σε αυτή την φάση.

Επίσης στην εργασία 2 πραγματοποιήσαμε διορθώσεις και βελτιώσεις στους αλγορίθμους πολυγωνοποίησης. Δεν πραγματοποιήσαμε κάποια αλλαγή στους αλγόριθμους βελτιστοποίησης. Οπότε δεν έχει γίνει κάποια αλλαγή στα παλαιότερα αρχεία. Απλώς προστέθηκαν τα αρχεία της εργασίας 3 που αναφέραμε παραπάνω.

evaluate.cpp:

Επιλέξαμε να χρησιμοποιήσουμε και να αξιολογήσουμε τους αλγορίθμους incremental και convex hull με local search (L=10, threshold=0.0001) και simulated annealing (L=10000, global step) για ελαχιστοποίηση και μεγιστοποίηση. Συνολικά προκύπτουν οι 8 συνδυασμοί:

```
"inc_ls_min"
"inc_ls_max"
"inc_siman_min"
"inc_siman_max"
"ch_ls_min"
"ch_ls_max"
"ch_siman_min"
"ch_siman_max"
```

Η εφαρμογή λειτουργεί ως εξής:

- Την εκτελούμε με την εντολή: ***./evaluate -i <directory_with_files> -o <output_file>***
- Σώζουμε όλα αρχεία .instance του φακέλου εισόδου σε ένα vector
- Φτιάχνουμε ένα unordered_map και το γεμίζουμε με στοιχεία που ως κλειδί έχουν το μέγεθος των αρχείων και ως τιμή ένα vector με τα αρχεία του αντίστοιχου μεγέθους.
- Για κάθε μέγεθος, για κάθε αρχείο του μεγέθους, για κάθε αλγόριθμο:
    - Δημιουργούμε μια διεργασία παιδί
    - Καλούμε την ualarm με τον χρόνο cut-off (500*n*1000 microseconds).
    - Εκτελούμε τον αλγόριθμο στο αρχείο.
    - Η μητρική διεργασία περιμένει να επιστρέψει το score το παιδί
    - Εισάγουμε το score σε ένα unordered_map με ως κλειδί έχουν το μέγεθος των αρχείων και ως τιμή ένα vector με τα score του αντίστοιχου μεγέθους.
- Για κάθε μέγεθος, για κάθε αλγόριθμο:
    - Βρίσκουμε τα min score, max score, min bound, max bound από τα scores που αποθηκεύσαμε στο unordered_map
    - Τα εκτυπώνουμε στο αρχείο εξόδου με κατάλληλη μορφοποίηση.

includes/evaluate_helpers/evaluate.hpp:

Περιέχει τις συναρτήσεις:

***exec*:** εκτελεί μια εντολή shell και επιστρέφει το αποτέλεσμά της ως string.

find: αναζητά αν υπάρχει ένα string σε ενα vector από strings.

print_correct_use: εκτυπώνει την σωστή χρήση της evaluate.

process_input: εξάγει τον φάκελο εισόδου και το αρχείο εισόδου από τα ορίσματα της εκτέλεσης.

get_size_of_file: εξάγει το μέγεθος του αρχείου από την ονομασία ενός αρχείου με μορφή file_name-xxxxxxx.instance.

get_alg_scores: εξάγει τα score από την εκτέλεση ενός αλγορίθμου από το vector με scores που σώζονται στο unordered_map μετά την εκτέλεση κάθε αλγορίθμου.

vec_sum: αθροίζει τα στοιχεία ενός vector..

vec_min: βρίσκει το ελάχιστο από τα στοιχεία ενός vector.

vec_max: βρίσκει το μέγιστο από τα στοιχεία ενός vector.