Ηλιάνα-Χάρις Σιούστη 1115201100107

Κουγιανός Νικόλαος 1115201100060

15/1/2017

Ανάπτυξη Λογισμικού για

Αλγοριθμικά Προβλήματα

3η Προγραμματιστική Εργασία

**Υλοποίηση αλγορίθμων υπόδειξης (Recommendation) & Συσταδοποίηση μοριακών διαμορφώσεων**

στη γλώσσα C

**1ο Σκέλος Εργασίας**

Αρχεία Κώδικα:

previousFunctions.c: Περιέχει τις συναρτήσεις που χρησιμοποιούμε από την εργασία 2.

ergasia3.c: Περιέχει την κύρια συνάρτηση του προγράμματος, τη συνάρτηση διαβάσματος αρχείου εισόδου, καθώς και τη συνάρτηση clustering recommendation

Από τους δυνατούς συνδυασμούς initialization – assignment - update των συναρτήσεων που είχαν υλοποιηθεί στην εργασία 2, επιλέχθηκε ο εξής: concentrate – pam\_assigmenet - lloyds.

Υπάρχουν έλεγχοι αποτυχίας δέσμευσης μνήμης και τερματίζεται το πρόγραμμα στην περίπτωση αποτυχίας. Σε αντίθετη περίπτωση, γίνεται αποδέσμευση της μνήμης πριν την έξοδο του προγράμματος.

Το πρόγραμμα είναι πλήρως παραμετρικό.

Έχει υλοποιηθεί πλήρως το Β μέρος Μέθοδος συσταδοποίησης (Clustering) Recommendation, ενώ δεν έχει υλοποιηθεί το Α μέρος και το 10-foldcross-validation.

Περιγραφή Συναρτήσεων:

Αρχείο previousFunctions.c:

* double hammingDistance(double \*item1, double \*item2, int dimension)

Έχουν γίνει κάποιες αλλαγές σε σχέση με τις προηγούμενες εργασίες ούτως ώστε οι βαθμολογίες από 2.5 και πάνω να θέτονται ως ένα και από 2.5 και κάτω ως 0.

* double eucledianDistance(double \*item1, double \*item2, double avg1, double avg2, int dimension) & double cosineDistance(double \*item1, double \*item2, double avg1, double avg2, int dimension)

Έχουν γίνει κάποιες αλλαγές σε σχέση με τις προηγούμενες εργασίες ούτως ώστε να λαμβάνονται υπόψιν και ο μέσος όρος των βαθμολογιών.

Αρχείο ergasia3.c:

* void ParseFile(char\* inputFile, int\* userCount, int\* movieCount, int\* p, int\* userGivesP)

Δέχεται το αρχείο εισόδου και βρίσκει τον αριθμό των χρηστών, τον αριθμό των ταινιών καθώς και αν δίνεται το p ή όχι.

* void fillMovieRatings(char\* inputFile, double\*\* array, int userCount, int movieCount, int userGivesP)

Γεμίζει τον πίνακα movieRatings με τις βαθμολογίες των ταινιών.

* void fillAvgRatingsPerUser(double \*\*movieRatings, double \*array, int userCount, int movieCount)

Υπολογίζεται ο μέσος όρος των βαθμολογιών των ταινιών που κρατώνται στον πίνακα avgRatingsPerUser.

* void clustering\_recommendation(double \*\*movieRatings, int userCount, int movieCount, double\* avgRatingsPerUser, char distance\_metric, int P, int argc)

Για την εύρεση του βέλτιστου Κ, ως αρχική τιμή δίνεται το n/(p-2) και φτάνουμε μέχρι το n/(p+3), λόγω του ότι είναι χρονοβόρα διαδικασία αν και έχει ελεγχθεί για διάφορες τιμές του K. Επιπλέον, ο αλγόριθμος Lloyds κάνει 3 loops για τον ίδιο λόγο.

Οδηγίες μεταγλώττισης :

/\* Makefile \*/

OBJS = ergasia3.o previousFunctions.o

SOURCE = ergasia3.c previousFunctions.c

HEADER = previousFunctions.h

OUT = ergasia3

CC = gcc

CFLAGS = -g -c

all: $(OBJS)

$(CC) -g $(OBJS) -o $(OUT) -lm

ergasia3.o: ergasia3.c

$(CC) $(CFLAGS) ergasia3.c

previousFunctions.o: previousFunctions.c

$(CC) $(CFLAGS) previousFunctions.c

clean:

rm -f $(OBJS) $(OUT)

count:

wc $(SOURCE) $(HEADER)

Το πρόγραμμα μεταγλωττίζεται με τη χρήση της εντολής:

make -f Makefile

**2ο Σκέλος Εργασίας**

Αρχεία Κώδικα:

ClusterFunctions.c: Περιέχει τις υλοποιήσεις των ζητούμενων αλγορίθμων,

παράλληλα με κάποιες βοηθητικές συναρτήσεις.

Bio.c: Περιέχει συναρτήσεις που είναι υπεύθυνες για το διάβασμα του αρχείου εισόδου, καθώς και τη συνάρτηση που χρησιμοποιείται για τη μετρική d-RMSD.

main.c: Περιέχει τη συνάρτηση που είναι υπεύθυνη για την κλήση της κύριας

συνάρτησης και τη διάδραση με τον χρήστη.

Περιγραφή Συναρτήσεων:

Αρχείο ClusterFunctions.c:

* int sameMedoids(int \*oldMedoids, int \*newMedoids, int k)

Συνάρτηση που ελέγχει κατά πόσο δυο πίνακες ακεραίων (ίσων στοιχείων) είναι ίδιοι. Δέχεται σαν παραμέτρους δυο πίνακες ακεραίων και το πλήθος των στοιχείων τους (όχι αθροιστικά) και επιστρέφει 1 σε περίπτωση που τα στοιχεία των πινάκων είναι ίδια, 0 σε διαφορετική περίπτωση.

* int\* initializationConcentrate(float \*\*distanceMatrix, int itemCount, int k)

Συνάρτηση που υπολογίζει τα αρχικά medoids για την αρχικοποίηση (initialization) των clusters σύμφωνα με τον αλγόριθμο Initialization Concentrate. Δέχεται ως παραμέτρους τον πίνακα αποστάσεων των στοιχείων, το πλήθος τους, καθώς και το πλήθος των medoids και επιστρέφει έναν πίνακα που περιέχει τις θέσεις των στοιχείων που επιλέχθηκαν ως medoids στον αρχικό πίνακα (και κατ’ επέκταση και στον πίνακα αποστάσεων). Αρχικά, υπολογίζεται ένας πίνακας που περιέχει το άθροισμα των αποστάσεων των στοιχείων από τα υπόλοιπα (παρονομαστής του κλάσματος) και ένας πίνακας που περιέχει τα τετράγωνα των αποστάσεων

των στοιχείων από τα υπόλοιπα (αριθμητής του κλάσματος) και παράλληλα υπολογίζεται το εκάστοτε vi. Επίσης δημιουργείται και ένας πίνακας που περιέχει τον αύξοντα αριθμό του κάθε στοιχείου στον αρχικό πίνακα. Αφού τελειώσει αυτή η διαδικασία, γίνεται χρήση της συνάρτησης SelectionSortTwoArrays για την παράλληλη ταξινόμηση του πίνακα που περιέχει τις τιμές vi για το κάθε στοιχείο και του πίνακα που περιέχει τον αύξοντα αριθμό των στοιχείων. Τελικά, επιλέγονται τα στοιχεία με τις k μικρότερες τιμές vi σε έναν πίνακα ακεραίων.

* void SelectionSortTwoArrays(double \*array, int \*array2, int arrayLength)

Συνάρτηση που υλοποιεί τον αλγόριθμο της selection sort. Η ταξινόμηση γίνεται βάσει του πρώτου πίνακα (array) και παράλληλα ταξινομούνται (αλλάζουν θέση) και οι τιμές του δεύτερου πίνακα (array2).

* int\* assignmentPAM(float \*\*distanceMatrix, int \*medoids, int itemCount, int k, double \*totalCost)

Συνάρτηση που προσομοιώνει την καταχώρηση των αντικειμένων στα αντίστοιχα clusters που πρέπει να πάνε βάσει του αλγορίθμου PAM (simple approach). Δέχεται σαν παραμέτρους τον πίνακα αποστάσεων των αντικειμένων, έναν πίνακα με τους αύξοντες αριθμούς (σαν στοιχεία του αρχικού πίνακα) των εκάστοτε medoids, το πλήθος των αντικειμένων, το πλήθος των medoids και το συνολικό κόστος για το εκάστοτε clustering, και επιστρέφει έναν πίνακα ακεραίων με τον αύξοντα αριθμό των medoids (αυτή τη φορά βάσει του πίνακα int \*medoids). Για κάθε στοιχείο, επιλέγουμε ένα μεγάλο αριθμό σαν την ελάχιστη απόσταση από ένα medoid και για κάθε medoid ελέγχουμε τον πίνακα αποστάσεων για να βρούμε τελικά από ποιό medoid έχει την μικρότερη απόσταση και να το καταχωρήσουμε στο cluster που

αντιπροσωπεύει εκείνο το medoid. Παράλληλα υπολογίζεται και το συνολικό κόστος του clustering.

* int\* updateClarans(float \*\*distanceMatrix, int \*clustersArray, int \*medoids, int itemCount, int k, double \*totalCost, int numLocal, int maxNeighbor)

Συνάρτηση ολοκληρώνει την διαδικασία του clustering, βρίσκει τα βέλτιστα medoids βάσει του αλγορίθμου CLARANS (update) και αναλαμβάνει και το βήμα του assignment (ανάλογα με τη μέθοδο που έχει επιλεχθεί) μέχρι να τερματίσει η λειτουργία της. Δέχεται ως παραμέτρους τον πίνακα αποστάσεων των σημείων, τον πίνακα με τα clusters(medoids) το οποίου είναι μέλος κάθε σημείο, τον πίνακα με τα medoids, το πλήθος αυτών, το συνολικό κόστος, τον τύπο του assignment και το πλήθος των επαναλήψεων του αλγορίθμου ,και επιστρέφει τον τελικό πίνακα με τα medoids, στα clusters των οποίων είναι μέλος κάθε σημείο. Αρχικά θέτουμε ως τοπικό μέγιστο κόστος αυτό που βρήκε το βήμα του assignment, πριν ξεκινήσει η διαδικασία του update. Για καθεμία από τις numLocal φορές που του δίνονται ως είσοδος, διαλέγει, αρχικά και τυχαία Κ items ως αρχικά medoids και αρχικοποιεί μια μεταβλητή (j) στο 0. Έπειτα, τυχαία διαλέγει ένα από αυτά και το ανταλλάσει με ένα μη-medoid (του cluster του) και αφού κάνει assignment (με όποια μέθοδο έχει επιλεγεί στο 2ο βήμα – παράμετρος και αυτή), υπολογίζει το συνολικό κόστος. Αν το κόστος συμφέρει (μικρότερο), σε σχέση με το προηγούμενο, κρατάει την αλλαγή αυτή και ο μετρητής (j) ξεκινάει πάλι από το 0. Αν όχι, επαναφέρει την προηγούμενη κατάσταση και αυξάνει τον μετρητή αυτόν κατά 1. Όταν ο μετρητής αυτός (j) φτάσει την τιμή maxNeighbor, έχουμε ολοκληρώσει ένα loop, βρίσκοντας ένα τοπικό ελάχιστο, το οποίο το συγκρίνουμε με το καλύτερο τοπικό έως τώρα και κρατάμε

το καλύτερο. Αφού επαναληφθεί numLocal φορές, το καλύτερο τοπικό ελάχιστο το θεωρούμε τελικό (ολικό ελάχιστο). Οπότε, αυτή η συνάρτηση επιστρέφει, τα τελικά Κ medoids, την τελική ανάθεση (assignment) των items στα clusters και το συνολικό κόστος.

* void medoidDistance(float \*\*distanceMatrix, double \*minDistance, double \*maxDistance, int \*medoids, int k)

Συνάρτηση που υπολογίζει την μέγιστη και ελάχιστη απόσταση μεταξύ των medoids (ζευγάρια ανά δυο). Δέχεται σαν παραμέτρους τον πίνακα αποστάσεων μεταξύ των σημείων, την ελάχιστη και μέγιστη απόσταση μεταξύ των medoids.

* double\* silhouette(float \*\*distanceMatrix, int \*clustersArray, int \*medoids, int itemCount, int k)

Συνάρτηση που υπολογίζει την μετρική αξιολόγησης βάσει του αλγορίθμου Silhouette. Δέχεται ως παραμέτρους τον πίνακα αποστάσεων των στοιχείων, τον πίνακα που περιέχει τον αύξοντα αριθμό των medoids στων οποίων τα clusters είναι μέλη τα στοιχεία, τον πίνακα με τον αύξοντα αριθμό των medoids, τον πλήθος των στοιχείων και το πλήθος των medoids και επιστρέφει έναν πίνακα με τα αποτελέσματα της αξιολόγησης, τόσο για κάθε cluster, όσο και για το clustering συνολικά. Αρχικά υπολογίζεται το άθροισμα των μέσων αποστάσεων των στοιχείων

από τα υπόλοιπα στοιχεία του cluster και ελέγχονται τα γειτονικά clusters για να βρεθεί το γειτονικό cluster. Στη συνέχεια, υπολογίζεται το άθροισμα των μέσων αποστάσεων των στοιχείων από τα στοιχεία του γειτονικού cluster και τελικά υπολογίζεται και επιστρέφεται τα αποτελέσματα της αξιολόγησης.

Αρχείο Bio.c:

* float\*\* getData(char \*fileName, int \*conformationCount, int \*N)

Συνάρτηση που διαβάζει το αρχείο εισόδου και προσωρινά αποθηκεύει τα δεδομένα στη μνήμη. Δέχεται ως παραμέτρους το όνομα του αρχείου εισόδου, το πλήθος των

διαμορφώσεων και το πλήθος των τριάδων της κάθε διαμόρφωσης και επιστρέφει έναν δισδιάστατο πίνακα, ο οποίος σε κάθε του γραμμή περιέχει όλες τις N τριάδες της κάθε διαμόρφωσης. Αρχικά, η συνάρτηση ανοίγει το αρχείο εισόδου και διαβάζει τις 2 πρώτες γραμμές που περιέχουν το πλήθος των διαμορφώσεων και το πλήθος των τριάδων κάθε διαμόρφωσης και τα καταχωρεί στις μεταβλητές που παρέχονται ως παράμετροι στη συνάρτηση. Έπειτα, καταχωρεί τις N τριάδες για κάθε διαμόρφωση στην κατάλληλη θέση στον δισδιάστατο πίνακα και τελικά κλείνει το αρχείο και επιστρέφει τον δισδιάστατο πίνακα.

* float getdRMSD(float \*conf1, float \*conf2, int confSize)

Συνάρτηση που υπολογίζει τη μετρική d-RMSD για δύο δοσμένες διαμορφώσεις. Δέχεται ως παραμέτρους τις Ν τριάδες κάθε διαμόρφωσης, καθώς και το μέγεθος αυτών (δηλαδή, το πλήθος των τριάδων κάθε διαμόρφωσης) και επιστρέφει το αποτέλεσμα που δίνει η μετρική d- RMSD.

Αρχείο main.c:

* int main(int argc, const char \* argv[])

Η κύρια συνάρτηση του προγράμματος που είναι υπεύθυνη για τη συσταδοποίηση των διαμορφώσεων και την εύρεση του βέλτιστου πλήθους συστάδων. Αρχικά, η συνάρτηση ανοίγει το αρχείο και καλεί την συνάρτηση getData για να διαβάσει τα δεδομένα από το αρχείο. Στη συνέχεια, κατασκευάζει τον πίνακα αποστάσεων των διαμορφώσεων καλώντας τη συνάρτηση getdRMSD επαναληπτικά. Έπειτα, καλούνται οι συναρτήσεις για τη συσταδοποίηση των δεδομένων ώστε να βρεθεί το βέλτιστο πλήθος συστάδων για το δοσμένο dataset, και αφού αυτό βρεθεί, καλούνται οι συναρτήσεις συσταδοποίησης εκ νέου, δημιουργείται το αρχείο εξόδου και γράφονται σε αυτό το βέλτιστο πλήθος συστάδων που βρέθηκε, το αποτέλεσμα της μετρικής Silhouette, καθώς και τα στοιχεία που βρίσκονται σε κάθε συστάδα.

Τελικά, κλείνει το αρχείο εξόδου και αποδεσμεύει τη μνήμη.

Λόγω ελλιπούς κατανόησης της χρήσης των μαθηματικών βιβλιοθηκών που προτάθηκαν και έλλειψης χρόνου, δεν χρησιμοποιήθηκε η μετρική c-RMSD, αλλά η d-RMSD, σε μια προσπάθεια να υλοποιηθούν τα ζητούμενα της εργασίας.

Οδηγίες μεταγλώττισης :

/\* Makefile \*/

OBJS = main.o ClusterFunctions.o Bio.o

SOURCE = main.c ClusterFunctions.c Bio.c

HEADER = ClusterFunctions.c Bio.h

OUT = bio

CC = gcc

CFLAGS = -g -c

all: $(OBJS)

$(CC) -g $(OBJS) -o $(OUT) -lm

main.o: main.c

$(CC) $(CFLAGS) main.c

ClusterFunctions.o: ClusterFunctions.c

$(CC) $(CFLAGS) ClusterFunctions.c

Bio.o: Bio.c

$(CC) $(CFLAGS) Bio.c

clean:

rm -f $(OBJS) $(OUT)

count:

wc $(SOURCE) $(HEADER)

Το πρόγραμμα μεταγλωττίζεται με τη χρήση της εντολής:

make -f Makefile