ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Ανάπτυξη εφαρμογής για τον υπολογισμό ερωτήσεων κορυφογραμμής με βάση την απόσταση

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΑΘΑΝΑΣΙΟΥ ΕΞΙΖΟΓΛΟΥ & ΑΪΒΑΤΙΔΗ ΠΡΟΔΡΟΜΟΥ

Επιβλέπων : Βλάχου Ακριβή, Αναπληρώτρια Καθηγήτρια

Μέλη εξεταστικής επιτροπής: **Κωνσταντίνου Ελισάβετ, Αναπληρώτρια Καθηγήτρια**
Κωστούλας Θεόδωρος, Αναπληρωτής Καθηγητής

Σάμος, Σεπτέμβριος 2023

Η σελίδα αυτή είναι σκόπιμα λευκή.

Πρόλογος και ευχαριστίες

Η σύγχρονη εποχή διακρίνεται για την ταχεία εξέλιξη της τεχνολογίας και την αναγκαιότητα να εφαρμόσουμε καινοτόμες λύσεις σε διάφορους τομείς της ζωής μας. Ένας από αυτούς τους τομείς είναι η γεωγραφική πληροφορική, όπου η χρήση γεωχωρικών δεδομένων και τεχνικών επεξεργασίας τους έχει επιφέρει σημαντικές εξελίξεις. Η παρούσα πτυχιακή εργασία, με τίτλο "Ανάπτυξη εφαρμογής για τον υπολογισμό ερωτήσεων κορυφογραμμής με βάση την απόσταση" αποτελεί μια προσπάθεια να συνδυάσει τη γεωγραφία και την πληροφορική με σκοπό τη δημιουργία ενός εργαλείου που θα βελτιώσει την επεξεργασία γεωχωρικών δεδομένων. Στόχος αυτής της εργασίας είναι η ανάπτυξη μιας ευέλικτης εφαρμογής που θα επιτρέπει στους χρήστες να εκτελούν υπολογισμούς ερωτήσεων κορυφογραμμής με ακρίβεια και ταχύτητα, βασιζόμενη στην απόσταση ανάμεσα στα γεωγραφικά σημεία. Επιπλέον, η εφαρμογή αυτή θα προσφέρει δυνατότητες αναπαράστασης και οπτικοποίησης των αποτελεσμάτων, κάτι που μπορεί να επιτρέψει στους χρήστες να κατανοήσουν καλύτερα τα γεωγραφικά δεδομένα που διαχειρίζονται.

Θα θέλαμε να αποδώσουμε τις ευχαριστίες μας στην επιβλέπουσα καθηγήτρια Βλάχου Ακριβή για την καθοδήγηση και την επιτήρησή της κατά τη διάρκεια της υλοποίησης της εργασίας, όλους τους συγγραφείς των έργων που αναφέρονται σε αυτή την εργασία για την συνεισφορά τους στη γνώση και την έρευνα και τέλος ένα ευχαριστώ στην οικογένεια και τους φίλους μας για την ενθάρρυνση και την υποστήριξη που μας παρείχαν.

Πρόδρομο Αϊβατίδη, Αθανάσιο Εξίζογλου

Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

Η σελίδα αυτή είναι σκόπιμα λευκή.

Πίνακας περιεχομένων

Λίστα Πινάκων

Λίστα Αλγορίθμων

Λίστα Εικόνων

Ακρωνύμια

BNL	Block Nested Loop-αλγόριθμος υπολογισμού κορυφογραμμής
ms	Milisecond- Μονάδα μέτρησης χρόνου
Min	Minimum – Ελάχιστη τιμή
Max	Maximum – Μέγιστη τιμή
Buffer	Ενδιάμεση μνήμη προσωρινής αποθήκευσης του υπολογιστικού συστήματος
Int	Τύπος αριθμητικών ακέραιων τιμών στην Java
Float	Τύπος αριθμητικών δεκαδικών τιμών στην Java
Double	Τύπος αριθμητικών δεκαδικών τιμών μεγάλης χωρητικότητας στην Java
True	Μπουλιανή τιμή που δηλώνει αλήθεια στην Java
False	Μπουλιανή τιμή που δηλώνει αναλήθεια στην Java
Null	Τιμή που δηλώνει τη μη ύπαρξη αντικειμένου ή τιμής στην Java
Input	Εισροή
Output	Εκροή
For	Επαναληπτικός βρόγχος ενεργειών στην java
While	Επαναληπτικός βρόγχος ενεργειών στην java
If..else	Βρόγχος ελέγχου συνθήκης στην java
Query	Επερώτημα
Freq	Frequency- Συχνότητα
Java	Αντικειμενοστρεφής γλώσσα προγραμματισμού
Try..catch	Βρόγχος δοκιμής ενεργειών και διαχείρισης σφαλμάτων στην java
Add	Προσθήκη
Remove	Αφαίρεση
BBS	Branch and Bound Skyline Algorithm
NN	Nearest Neighbor Algorithm
Csv	Comma Separated Values- Τύπος αρχείων κειμένου με τον χαρακτήρα «,» ως διαχωριστικό τιμών
Open	Άνοιξε
Close	Κλείσε
Lat	Latitude- γεωγραφικό πλάτος
Long	Longtitude- γεωγραφικό μήκος
Std	Standard Deviation- Τυπική απόκλιση

VIN	Αριθμός οχήματος
No	Νούμερο
Ascending	Άυξουσα
Descending	Φθίνουσα
Ø	κενό
S_distance	Skyline distance – Απόσταση κορυφογραμμής

Περίληψη

Στο πρόβλημα το οποίο έρχεται μέσα από την ανάγκη του χρήστη για τον προσδιορισμό της βέλτιστης επιλογής σε μια μεγάλη πληθώρα επιλογών φέρει λύση η τεχνολογική ανάπτυξη των ερωτήσεων κορυφογραμμής. Η διακλάδωση αυτών των επιλογών συνήθως είναι ογκώδες και ξεπερνά την καθημερινή απλή και όχι εν τάχει σκέψη του.

Μέσα από την εν λόγω διπλωματική διατριβή υλοποιείται εφαρμογή, η οποία είναι ικανή να δέχεται μεγάλη ποσότητα πληροφορίας, μπορεί να προσαρμόζεται με βάση τις προτιμήσεις κριτηρίων πάνω σε συγκεκριμένα αντικείμενα που θέλει ο χρήστης να εστιάσει και του προσφέρει τις βέλτιστες επιλογές που θα μπορεί να επιλέξει. Εφόσον γίνει αναλυτική περιγραφή πάνω σε βασικά θέματα, τα οποία συσχετίζονται με τις ερωτήσεις κορυφογραμμών, μελέτη πάνω σε θεμελιώδεις αλγορίθμους και σύγκριση μεταξύ τους, ιστορική αναδρομή και βιβλιογραφική ανασκόπηση επί του θέματος, ακολουθεί και πειραματική μελέτη της εφαρμογής. Η εφαρμογή δέχεται μεγάλο πλήθος εγγραφών και εμφανίζει την κορυφογραμμή για τον χρήστη, η οποία είναι βασισμένη στον παράγοντα απόσταση από ένα σημείο απόστασης. Ως παράδειγμα έχουν χρησιμοποιηθεί πραγματικά δεδομένα από πωλήσεις αυτοκινήτων. Επιπλέον συγκρίνεται η απόδοση της εφαρμογής με μία αφελή προσέγγιση του αλγορίθμου.

Τέλος, προκύπτουν συμπεράσματα για την απόδοση της εφαρμογής και τη μελέτη της διπλωματικής διατριβής.

Λέξεις Κλειδιά: *Ερωτήματα κορυφογραμμής, Χωρικά δεδομένα, Σύστημα υποστήριξης λήψης αποφάσεων*

Abstract

The technological development of top-line questions offers a solution to the problem that arises from the user's need to determine the optimal choice from a wide array of options. The branching of these choices is usually voluminous and goes beyond the everyday, not hastily considered thought process.

Through this dissertation, an application is implemented that is capable of handling a large amount of information, can adapt based on the user's preferences and criteria on specific items of interest, and provides the best options for the user to choose from. Following a detailed description of fundamental topics related to top-line questions, a study on fundamental algorithms and a comparison between them, a historical retrospective, and a literature review on the subject, an experimental study of the application is conducted.

The application receives a large number of records and displays the top line for the user, which is based on the distance factor from a reference point. Real data from car sales have been used as an example. Furthermore, the application's performance is compared with a naive algorithm approach.

In conclusion, findings are drawn regarding the application's performance and the study of the dissertation.

Keywords: *Skyline Queries, Spatial Data, Decision-making support system*

1 *Εισαγωγή*

1.1 Εισαγωγή στις ερωτήσεις κορυφογραμμών

Πληθώρα είναι η πληροφορία καθώς και πολλές είναι οι αναζητήσεις που μπορούν να πραγματοποιηθούν μέσα σε μια βάση δεδομένων. Μέσα από αυτές τις αναζητήσεις ο χρήστης λαμβάνει κενές απαντήσεις και συνεπώς δεν πληρούνται όλες οι απαιτήσεις του, φτάνοντας σε ένα μη ικανοποιητικό επίπεδο. Σε αυτό το σημείο ο χρήστης διασταυρώνεται με το να θέλει να αναζητήσει την καλύτερη πτήση που θέλει να επιλέξει για ένα δικό του ταξίδι, ένα καινούργιο σπίτι που θέλει να νοικιάσει ή να αγοράσει, ακόμα και το να αποκτήσει ένα καινούργιο ή μεταχειρισμένο αυτοκίνητο. Οι κλασικές παραδοσιακές μηχανές βάσεων δεδομένων μπορούν αποτελεσματικά να εξυπηρετούν τον χρήστη όσον αφορά την αποθήκευση, την τροποποίηση και τη διαγραφή αρχείων, πληροφοριών και άλλων δεδομένων, αφήνοντας όμως ένα κομμάτι άδειο και αναπάντητο. Το κομμάτι της προτίμησης και σίγουρα το πρόβλημα γίνεται μεγαλύτερο όταν σε μία βάση δεδομένων τα στοιχεία τα οποία είναι αποθηκευμένα έχουν πολλαπλές διαστάσεις-παραμέτρους και απλά ερωτήματα κορυφογραμμών δεν μπορούν να δώσουν ένα αποτελεσματικό και βέλτιστο σενάριο για τον χρήστη. Ένα παράδειγμα του τι είναι κορυφογραμμή αρκετά γενικευμένα, για την περισσότερη κατανόηση του φαίνεται ως εξής: Ένας χρήστης ο οποίος στην προσωπική του βάση δεδομένων έχει συλλέξει δεδομένα διαφόρων και αγαπημένων αυτοκινήτων με σκοπό την αγορά κάποιου συγκεκριμένου. Καθένα από αυτό το σύνολο αυτοκινήτων έχει δύο διαστάσεις σαν παραμέτρους, την τιμή και την χρονολογία του. Μία ερώτηση κορυφογραμμών απομονώνει το σύνολο έχοντας πλειάδες όπως ένα TOYOTA [15000,1999], ένα PEUGEUT [18000,2004] και τέλος ένα AUDI [15000,2003]. Αν η ερώτηση όσον αφορά το βέλτιστο αυτοκίνητο, έχει ως κριτήριο την χαμηλή τιμή αγοράς του, τότε η κορυφογραμμή θα επιστρέψει ως έξοδο το φθηνότερο αυτοκίνητο, συνεπώς το TOYOTA [15000,1999]. Η κορυφογραμμή αλλάζει και διαμορφώνεται αντίστοιχα με τα κριτήρια εισόδου του χρήστη αναφερόμενα στις διαστάσεις των στοιχείων. Στο συγκεκριμένο παράδειγμα αν ο χρήστης θελήσει να κάνει μια ερώτηση κορυφογραμμής με κριτήριο την αύξουσα τιμή του έτους του αυτοκινήτου, τότε ο αλγόριθμος θα επέστρεφε το νεότερο αυτοκίνητο, PEUGEUT [18000,2004]. Από την άλλη πλευρά του νομίσματος οι προτιμήσεις διαφέρουν καθώς πρέπει να υπάρχει η επιλογή της ευελιξίας και της ποικιλίας. Μία κορυφογραμμή στην εφαρμογή αναζήτησης όπως τιμή αγοράς χαμηλή και έτος κυκλοφορίας υψηλό, θα είχε ως αποτέλεσμα την επιστροφή δύο αυτοκινήτων. Το TOYOTA δεν θα ήταν ένα από τα αυτοκίνητα το οποίο θα έβγαινε στην

έξοδο καθώς κυριαρχείται από τα άλλα δύο αυτοκίνητα, το PEUGEOT και το AUDI. Η κοστολόγηση μιας κορυφογραμμής συγκριτικά με άλλους φυσικούς χειριστές εύρεσης βέλτιστης προτίμησης διαφέρει και καθιστά μεγαλύτερη πρόκληση καθώς η κορυφογραμμή μιλάει συχνά στον επεξεργαστή. Αν στον υπολογιστή υπάρχει αρκετή διαθέσιμη μνήμη τότε η κορυφογραμμή είναι ικανή να μπορέσει να υπολογίσει όλα τα δεδομένα που υπάρχουν στον σύνολο με μόνο μία ερώτηση κορυφογραμμής για να επιτευχθεί η σάρωση δεδομένων. Είναι βέβαια αξιοσημείωτο ότι η επεξεργασία των δεδομένων στην κάθε πλειάδα, στην μνήμη του υπολογιστή, απαιτεί μεγάλο χρονικό διάστημα, εφόσον εφαρμόζεται κάθε φορά σύγκριση μεταξύ των βέλτιστων πλειάδων της τότε στιγμής.

Συνοπτικά, οι ερωτήσεις κορυφογραμμών (Skyline queries) είναι ερωτήματα σε μια βάση δεδομένων που αποσκοπούν στην εύρεση των κορυφαίων αποτελεσμάτων σε ένα σύνολο δεδομένων, με βάση ένα ή περισσότερα κριτήρια. Στην ουσία, ένα Skyline query επιστρέφει τα στοιχεία που δεν κυριαρχούνται από τα υπόλοιπα μέσα σε ένα σύνολο αποτελεσμάτων. Τέλος, οι ερωτήσεων κορυφογραμμών είναι συχνά χρήσιμες για την ανάλυση των δεδομένων, όπου τα αποτελέσματα που παρέχονται από τα παραδοσιακά ερωτήματα δεν είναι αρκετά σαφή ή δεν αποκαλύπτουν τις καλύτερες επιλογές στο σύνολο των δεδομένων. Υπάρχουν διάφορες παραλλαγές των ερωτήσεων κορυφογραμμών, οι οποίες μπορούν και δίνουν λύση σε αυτό το πρόβλημα και θα αναλυθούν πιο διεξοδικά στο δεύτερο κεφάλαιο της παρούσας διπλωματικής.

1.2 Χωρικά δεδομένα

Έναν σημαντικό ρόλο παίζουν και τα σημειακά χωρικά δεδομένα τα οποία αναφέρονται σε πληροφορίες πάνω σε τοποθεσίες ή ακόμα και στην εύρεση μιας συγκεκριμένης οντότητας, όπου θα μπορούσε να είναι ένας άνθρωπος, ένα αντικείμενο αλλά και ένα γεωγραφικό σημείο. Αυτά τα δεδομένα μπορούν να αναφέρονται σε διάφορες μορφές, όπως είναι το γεωγραφικό πλάτος και μήκος, διευθύνσεις, μέρη, ονόματα δρόμων, ονόματα βουνών, λιμνών, ποταμών καθώς επίσης θέσεις από GPS, κλπ.

Σε διάφορους τομείς όπως στη γεωπληροφορική, την επιστήμη και τεχνολογία της πληροφορικής, τη διαχείριση καταστημάτων λιανικής, την υπηρεσία παράδοσης τροφίμων, την ασφάλεια, τα σημειακά χωρικά δεδομένα έχουν συμβάλει σε σημαντικό βαθμό στην ανάπτυξη τους. Είναι γνωστό επίσης ότι τα δεδομένα αυτά χρησιμοποιούνται για να βελτιώσουν τις υπηρεσίες που παρέχονται στο κοινό ή για να αναλύσουν δεδομένα σε μεγάλη κλίμακα προκειμένου να προβλεφθούν τάσεις και να ληφθούν δυσνόητες αποφάσεις, κάτι το οποίο απασχολεί την συγκεκριμένη διπλωματική. Επιπλέον τα σημειακά χωρικά δεδομένα, με τις ακριβείς πληροφορίες που παρέχουν, είναι αρκετά χρήσιμα όσον αφορά την πλοήγηση, καθώς και στον σχεδιασμό διαδρομών και υπηρεσιών και διαδρομών τα οποία βασίζονται στις διευθύνσεις και τις τοποθεσίες. Ένα παράδειγμα για να κατανοήσουμε την χρησιμότητα των σημειακών χωρικών δεδομένων είναι μία χαρτογράφηση η οποία βασίζεται σε χρήσιμες και σημαντικές πληροφορίες όπως χώρες, ποτάμια, βουνά, συντεταγμένες, ονόματα περιοχών, διευθύνσεις, ονόματα επιχειρήσεων κλπ. Με την δημιουργία του χάρτη προσδίδεται χρήσιμη και καταλυτική πληροφορία σε ένα σύνολο ατόμων οι οποίοι αναζητούν είτε υπηρεσία, είτε γνώση. Είναι σημαντικό να σημειωθεί ότι τα σημειακά χωρικά δεδομένα μπορούν να

χρησιμοποιηθούν στο εύρος του GIS (Geographic Information System) για χαρτογράφηση και χωρική ανάλυση.

Επίσης με την πρόοδο της τεχνολογίας ο άνθρωπος κατάφερε να χρησιμοποιεί τα σημειακά δεδομένα για την παρακολούθηση και την πρόληψη φυσικών καταστροφών. Με έναν ιδιαίτερο τρόπο συμβάλουν στην παρακολούθηση φυσικών καταστροφών, όπως πλημμύρες, σεισμοί, και για την ανάπτυξη συστημάτων πρόληψης και αντίδρασης σε περιπτώσεις καταστροφών. Επιπλέον η χρησιμότητα τους μπορεί να βρεθεί και στον τομέα της τρισδιάστατης μοντελοποίησης και χαρτογράφησης, καθώς και ρομποτική γεωπληροφορική αλλά και στην διακίνηση δεδομένων σε πραγματικό χρόνο.

Εν κατακλείδι, η συμπερασματική διατύπωση των προαναφερθέντων πληροφοριών, είναι ότι τα σημειακά χωρικά δεδομένα, είναι χωρικά δεδομένα πάνω στην επιφάνεια της γης, τα οποία αντιπροσωπεύουν μία οντότητα, η οποία έχει ποικίλες μορφές, όπως μία τοποθεσία, ένα γεγονός στο χώρο, μια διεύθυνση URL ή ακόμα και μια ταχυδρομική κωδικοποίηση. Αυτή την τεράστια πληροφορία την διαχειρίζονται τα συστήματα πληροφορικής με πολλούς και διάφορους τρόπους.

1.3 Αντικείμενο διπλωματικής

Στο προηγούμενο κεφάλαιο δόθηκε μία εισαγωγή για την έννοια των ερωτήσεων κορυφογραμμών. Αν καθίσει κάποιος και αναλογιστεί την σημασία της κορυφογραμμής στην καθημερινότητα του ανθρώπου δίνοντας του αποτελέσματα σε πολυδιάστατα κριτήρια θα καταλάβει ότι η σημασία της κορυφογραμμής είναι αρκετά μεγάλη και χρήσιμη. Διλήμματα στην ζωή του ανθρώπου υπάρχουν πάντα, κάποια ίσως μεγαλύτερα και κάποια μικρότερα. Ένας καθημερινός άνθρωπος όταν έρχεται αντιμέτωπος με σενάρια προβληματισμού επιλογής, με δεδομένα που δεν μπορεί να τα συγκρίνει με το μυαλό του και οι επιλογές που θα ακολουθήσει θα είναι είτε να παρατήσει το πρόβλημα του, είτε να στραφεί στην επίλυση του προβλήματος του μέσω της βοήθειας του υπολογιστή. Ακόμα όμως και με την βοήθεια του υπολογιστή, η προτίμηση του δεν θα είναι ξεκάθαρη και βέλτιστη. Θα θέσει ένα ερώτημα στον υπολογιστή του όπου δεν θα μπορεί να του υποστηρίζει όλα τα κριτήρια του, στις διαστάσεις που θέλει και ιδιαίτερα όταν υπάρχει στο ερώτημα του και ο παράγοντας απόσταση.

Σκοπός αυτής της διπλωματικής είναι να ερευνήσει σε βάθος διάφορους παράγοντες οι οποίοι συνδέονται με τα ερωτήματα κορυφογραμμής με βάση την απόσταση, όπως είναι το μέγεθος ενός μεγάλου συνόλου δεδομένων, η πολυπλοκότητα των κριτηρίων που χρησιμοποιούνται για τον προσδιορισμό της κυριαρχίας και οι διαθέσιμοι υπολογιστικοί πόροι προς επεξεργασία. Φυσικά υπάρχουν μικρά σύνολα δεδομένων με μικρό αριθμό αντικειμένων με απλά κριτήρια και σε αυτή την περίπτωση τα ερωτήματα κορυφογραμμών μπορούν εύκολα να εκτελεστούν με την χρήση απλών μικρών αλγορίθμων. Ωστόσο για μεγάλα σύνολα δεδομένων απαιτούνται πιο προηγμένοι αλγόριθμοι οι οποίοι εξασφαλίζουν αποτελέσματα με ακρίβεια και σε εύλογο χρονικό διάστημα. Ένας τέτοιος αλγόριθμος υλοποιείται στην παρούσα

διπλωματική διατριβή σε γλώσσα προγραμματισμού Java. Στα επόμενα κεφάλαια θα ακολουθήσει βιβλιογραφική διατριβή όσον αφορά τις ερωτήσεις κορυφογραμμών και των αλγορίθμων όπου βασίζονται πάνω σε αυτές. Ο αλγόριθμος που υλοποιείται είναι μία παραλλαγή του προηγμένου αλγόριθμου BNL (Block Nested Loop) μέσα από τον οποίο ο χρήστης θα παίρνει βέλτιστα αποτελέσματα σε παραλλαγές ερωτήσεων του με ή και χωρίς τον παράγοντα απόσταση. Όσον αφορά τα οφέλη της διπλωματικής διατριβής, μέσω ενός τέτοιου αλγορίθμου, απλοποιείται η λήψη των αποφάσεων, καθώς τα ερωτήματα Skyline παρέχουν μια σαφή και συνοπτική αναπαράσταση των πιο κυρίαρχων σημείων σε ένα σύνολο δεδομένων, διευκολύνοντας τη λήψη πολύ δύσκολων αποφάσεων. Επίσης, η χρήση πολλαπλών κριτηρίων, καθώς οι ερωτήσεις κορυφογραμμών επιτρέπουν τη χρήση πολλαπλών κριτηρίων για τον προσδιορισμό της κυριαρχίας, καθιστούν δυνατό τον υπολογισμό πολλαπλών παραγόντων κατά τη λήψη αποφάσεων. Ένα ακόμα όφελος είναι η ανάλυση των δεδομένων σε πραγματικό χρόνο. Επιτρέπεται η γρήγορη και ακριβής ανάλυση των δεδομένων κατά την ενημέρωση τους. Τέλος μειώνεται ο όγκος των δεδομένων που πρέπει να αποθηκευτούν και να αναλυθούν, καθώς λαμβάνονται υπόψη μόνο τα κυρίαρχα σημεία.

Συνοψίζοντας, αντικείμενο της διπλωματικής είναι να φέρει εις πέρας την υλοποίηση ενός τέτοιου αλγορίθμου, ώστε να μπορεί να φέρει εξοικονόμηση χρόνου και κόπου επιστρέφοντας τα πιο βέλτιστα και αποδοτικά αποτελέσματα σε προβλήματα επιλογής του ανθρώπου.

1.4 Δομή της διπλωματικής

Στο πρώτο κεφάλαιο γίνεται εισαγωγή στις ερωτήσεις κορυφογραμμών με σκοπό την εμβάθυνση και την κατανόηση σε αυτές. Στο δεύτερο κεφάλαιο γίνεται επιπλέον μελέτη πάνω στις ερωτήσεις κορυφογραμμών, καθώς ακολουθεί παράδειγμα πάνω σε αυτές και ψευδοκώδικα για την προσομοίωση του. Σε αυτό το κεφάλαιο επίσης αναλύονται θεμελιώδης αλγόριθμοι, οι οποίοι επεξηγούνται και συγκρίνονται μεταξύ τους με σκοπό την εμφάνιση πλεονεκτημάτων και μειονεκτημάτων τους. Στο κεφάλαιο τρία γίνεται εκτενής αναφορά στην βιβλιογραφική ανασκόπηση περιέχοντας σημαντικές ιστορικές αναδρομές και πρωταρχικούς αλγορίθμους. Το κεφάλαιο τέσσερα εμπεριέχει επεξηγήσεις απ' όλες τις κλάσεις που απαρτίζουν τον αλγόριθμο της εφαρμογής που υλοποιήθηκε στην παρούσα διπλωματική διατριβή καθώς επίσης ποια είναι η χρησιμότητα του GUI της εφαρμογής και τι είναι ικανό να εμφανίσει προς τον χρήστη. Στο πέμπτο κεφάλαιο πραγματοποιείται πειραματική μελέτη στην οποία η εφαρμογή δέχεται μεγάλο πλήθος εγγραφών αυτοκινήτων και εμφανίζει αποτελέσματα βέλτιστων επιλογών για τον χρήστη, τα οποία είναι βασισμένα στον παράγοντα απόσταση και συγκρίνονται με μία αφελή προσέγγιση του αλγορίθμου. Τέλος, στο έκτο κεφάλαιο ακολουθεί σύνοψη της διπλωματικής εργασίας καθώς επίσης και την επέκταση την οποία θα μπορούσε να ακολουθήσει, ενώ στο έβδομο κεφάλαιο λαμβάνουν χώρα οι βιβλιογραφικές αναφορές.

2 *Αλγόριθμοι και ερωτήσεις κορυφογραμμών*

2.1 Εισαγωγή

Τα τελευταία χρόνια μεγάλο μέρος της επιστημονικής κοινότητας ασχολείται με το πρόβλημα της έκδοσης ολοκληρωμένων αποτελεσμάτων σε οποιαδήποτε ερώτηση του χρήστη. Σε μια αναζήτηση του χρήστη ενδέχεται να υπάρχουν πολλά και διαφορετικά κριτήρια, με αποτέλεσμα να μην είναι τις περισσότερες φορές ευδιάκριτη η σειρά ή ο συνδυασμός των προτιμήσεων των κριτηρίων που επιθυμεί ο κάθε χρήστης. Έτσι, δημιουργείται η ανάγκη χρήσης μεθόδων που βοηθούν στην καλύτερη κατανομή και επεξεργασία των ερωτημάτων αυτών δίνοντας μια πιο σφαιρική απάντηση.

Έστω ότι βρίσκεστε σε διαδικασία έρευνας για να αγοράσετε το επόμενο αυτοκίνητό σας και ψάχνετε για ένα μεταχειρισμένο αυτοκίνητο με λίγα διανυθέντα χιλιόμετρα που να είναι φθηνό. Δυστυχώς, αυτοί οι δύο στόχοι αντιφάσκουν μεταξύ τους, καθώς τα μεταχειρισμένα αυτοκίνητα που έχουν λίγα χιλιόμετρα τείνουν να πουλιούνται ακριβότερα. Το σύστημα βάσης δεδομένων της εφαρμογής αγγελιών που χρησιμοποιείτε για την έρευνά σας δεν είναι σε θέση να αποφασίσει ποιο αυτοκίνητο είναι καλύτερο για εσάς, αλλά μπορεί τουλάχιστον να σας παρουσιάσει όλες τις ενδιαφέρουσες για εσάς αγγελίες μεταχειρισμένων αυτοκινήτων. Τα προτεινόμενα αυτοκίνητα είναι όλα αυτά που δεν είναι χειρότερα από άλλα αυτοκίνητα και στις δύο διαστάσεις. Αυτό το λέμε σύνολο από προτεινόμενα αυτοκίνητα της κορυφογραμμής (Skyline). Από το Skyline, μπορεί να πάρει κανείς την τελική του απόφαση, σταθμίζοντας έτσι τη δική του προσωπική προτίμηση για τιμή και διανυθέντα χιλιόμετρα. Η κορυφογραμμή ορίζεται ως εκείνα τα σημεία που δεν κυριαρχούνται από οποιοδήποτε άλλο σημείο. Ένα σημείο κυριαρχεί σε άλλο σημείο αν είναι όσο καλό ή καλύτερο σε όλες τις διαστάσεις και καλύτερο σε τουλάχιστον μία διάσταση. Καταλαβαίνει κανείς ότι ο υπολογισμός της κορυφογραμμής γίνεται με την κάθε μέρα όλο και πιο σημαντικός, μέσα από την ανάγκη των ανθρώπων να επεξεργαστούν μεγάλες βάσεις δεδομένων και να αναζητήσουν την σωστή επιλογή τους. Στο κεφάλαιο αυτό θα υπάρχει περισσότερη διευκρίνηση για την έννοια των κορυφογραμμών και των ερωτήσεων τους, καθώς επίσης και παραδείγματα για περισσότερη κατανόηση τους. Επίσης γίνεται αναλυτική αναφορά πάνω στους αλγορίθμους που έχουν βοηθήσει το Skyline για την υλοποίηση και την δομή του καθώς και διαδοχικοί πίνακες και ψευδοκώδικες με επεξήγηση αυτών των αποτελεσμάτων.

Έχουν προταθεί και αναπτυχθεί πολλοί αλγόριθμοι ερωτημάτων κορυφογραμμών και καθένας από αυτούς έχει ισχυρά σημεία αλλά και αδυναμίες. Στην αρχή του κεφαλαίου σημειώνεται για παράδειγμα αλγόριθμος «Διαίρει και βασίλευε» (Divide and Conquer), ένας κλασσικός αλγόριθμος ο οποίος διαχωρίζει το βασικό του πρόβλημα σε υπό προβλήματα και μετά επιλύει αυτά τα υπό προβλήματα μεμονωμένα. Ο αλγόριθμος Nearest Neighbor βασίζεται πάνω στον αλγόριθμο Διαίρει και Βασίλευε έχοντας ως σκοπό να μπορεί να ευρετηριάσει ένα σύνολο δεδομένων ενός χωροταξικού διαχωρισμού(π.χ. RTree, KDTree), όμως κατ' επέκταση να σημειωθεί ότι είναι ένας από τους πιο αποδοτικούς αλγορίθμους επειδή επιστρέφει τα σημεία της κορυφογραμμής με πολύ υψηλή ταχύτητα. Επίσης αλγόριθμοι οι οποίοι συνδέονται και αυτοί με

την παραλλαγή ερωτήσεων κορυφογραμμής είναι η αφελής προσέγγιση της κορυφογραμμής ο αλγόριθμος Branch and Bound Skyline και τέλος ο Block nested Loop.

2.2 Ερωτήσεις κορυφογραμμών

Με την ραγδαία τεχνολογική εξέλιξη αυτές τις μέρες, και με την επιθυμία του ανθρώπου να εξιχνιάσει πολλά προβλήματα τόσο στην καθημερινότητα του αλλά και πιο στοχευμένα, σε πολλούς κλάδους της επιστήμης και της μηχανικής, οι ερωτήσεις κορυφογραμμών έρχονται να παίξουν πρωταγωνιστικό ρόλο καθώς είναι το κλειδί ώστε να ανοίξει η πόρτα στην επίλυση του προβλήματος αυτού.

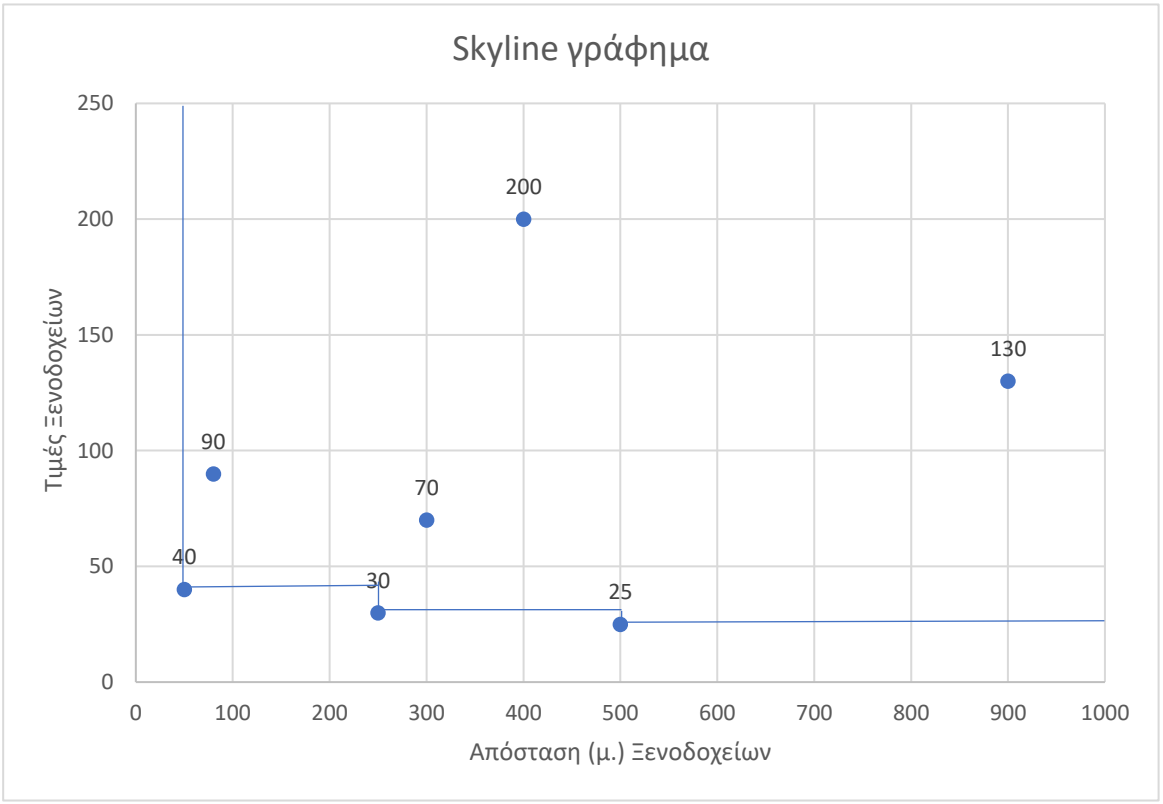
Η προσφορά των ερωτήσεων κορυφογραμμών, είναι η δυνατότητα εξερεύνησης και ανάκτησης των καλύτερων αποτελεσμάτων μια συλλογής στοιχείων βάση πολλαπλών κριτηρίων. Αντί να επιλέξει κάποιος μόνο ένα κριτήριο για την ταξινόμηση ή την επιλογή των αποτελεσμάτων, μέσω των ερωτήσεων κορυφογραμμών έχει την δυνατότητα να λάβει υπόψη πολλαπλά κριτήρια ταυτόχρονα και αυτό είναι που το καθιστά ιδιαίτερο. Η κύρια προσφορά των ερωτήσεων κορυφογραμμών είναι η απόκτηση όσο το δυνατόν καλύτερων αποτελεσμάτων, μέσα σε μία συλλογή με πολλαπλά στοιχεία και κατ' επέκταση πολλαπλά κριτήρια για το καθένα.

Πιο συγκεκριμένα οι ερωτήσεις κορυφογραμμών (Skyline queries) είναι ερωτήσεις οι οποίες πραγματοποιούνται για να επιστραφούν καλύτερα και πιο αποδοτικά αποτελέσματα βάση πολλαπλών κριτηρίων ώστε να λυθεί ένα πρόβλημα προτίμησης. Από επιστημονικής άποψης, η λογική της εφαρμογής τους δουλεύει ως εξής. Ένα στοιχείο α είναι κυρίαρχο κάποιου άλλου στοιχείου β, αν το στοιχείο α είναι καλύτερο ή ισοδύναμο σε όλες τις παραμέτρους του β και τουλάχιστον καλύτερο σε μία από αυτές. Όλα αυτά τα κυρίαρχα σημεία (Skyline points) που προκύπτουν δημιουργούν ένα σύνολο στοιχείων, μέσα από το οποίο μπορεί να γεννηθεί η κορυφογραμμή, η οποία θα φανερώσει στον χρήστη ένα βέλτιστο, αποδοτικό σενάριο προς επιλογή. Για να μπορέσει να γίνει πιο κατανοητή η έννοια και η σημασία των ερωτήσεων κορυφογραμμών αλλά και η ίδια η κορυφογραμμή ακολουθεί παράδειγμα.

Ας γίνει η υπόθεση ότι υπάρχει μία βάση δεδομένων με ξενοδοχεία και απαρτίζεται από τα εξής χαρακτηριστικά για κάθε ξενοδοχείο: Αριθμός εγγραφής , τιμή ανά διανυκτέρευση και τέλος, απόσταση από το κέντρο της πόλης. Για να εκτελεστεί ένα ερώτημα κορυφογραμμών για την αναζήτηση των βέλτιστων ξενοδοχείων θα χρησιμοποιηθούν οι δύο διαστάσεις από τα χαρακτηριστικά των ξενοδοχείων και αυτά είναι η τιμή ανά διανυκτέρευση και η απόσταση από τη πόλη. Μέσα από αυτή την ερώτηση, θα επιστρέψει ένα σύνολο ξενοδοχείων για τον χρήστη, τα οποία ξενοδοχεία είναι κυρίαρχα από τις υπόλοιπες εγγραφές στις δύο αυτές συγκεκριμένες διαστάσεις. Στο παρακάτω πίνακα αναγράφονται αναλυτικά οι αριθμοί εγγραφών, οι τιμές διανυκτερεύσεις και οι αποστάσεις από το κέντρο της πόλης. Η προτίμηση για την περάτωση της ερώτησης θα είναι τόσο για την τιμή όσο και για την απόσταση σε φθίνουσα σειρά.

ΞΕΝΟΔΟΧΕΊΟ	ΤΙΜΗ	ΑΠΟΣΤΑΣΗ(μ.)
1	25	500
2	30	250
3	130	900
4	40	50
5	70	300
6	200	400
7	90	80

Πίνακας 1: Απεικόνιση στοιχείων εγγραφών



Εικόνα 1: Γράφημα με σκοπό την ένδειξη κορυφογραμμής

Στο παραπάνω γράφημα εμφανίζεται το αποτέλεσμα της ερώτησης κορυφογραμμών. Στον άξονα y απεικονίζονται όλες οι τιμές των ξενοδοχείων ενώ στον οριζόντιο άξονα x φαίνονται όλες οι αποστάσεις των ξενοδοχείων από το κέντρο της πόλης. Οι κουκίδες, οι οποίες υπάρχουν μέσα στο γράφημα, έχουν πάρει θέση αναλογικά με τις τιμές από αυτές τις δύο διαστάσεις όπου περιγράφηκαν προηγουμένως, έχοντας σαν ετικέτα στο πάνω μέρος τους την τιμή διανυκτέρευσης. Παρατηρεί κανείς μέσα στο γράφημα ότι δημιουργείται μία γραμμή μέσω των κυρίαρχων σημείων (Skyline points) με αριθμό εγγραφής 4, 2 και 1. Αυτά τα ξενοδοχεία είναι κυρίαρχα των υπόλοιπων ξενοδοχείων στις δύο διαστάσεις τους ή τουλάχιστον σε μία από αυτές και με αυτόν τον τρόπο δίνει στον χρήστη την κορυφογραμμή, αποδίδοντας του ένα βέλτιστο και αποδοτικό σενάριο προς επιλογή.

Σε συνέχεια του παραδείγματος, μέχρι στιγμής έγινε μία προσέγγιση σε θεωρητικό υπόβαθρο για τον προσδιορισμό και την κατανόηση της έννοιας των κορυφογραμμών και των παραλλαγών ερωτήσεων πάνω σε αυτές. Όλα τα δεδομένα αποθηκεύονται μέσα σε μία βάση δεδομένων έχοντας ένα σύνολο οργανωμένο, προσβάσιμο και διαχειρίσιμο. Οι βάσεις δεδομένων διαθέτουν μια δομή που οργανώνει τα δεδομένα σε πίνακες. Κάθε πίνακας αποτελείται από στήλες και γραμμές, όπου κάθε στήλη αντιπροσωπεύει ένα χαρακτηριστικό ή πεδίο και η κάθε γραμμή αντιστοιχεί σε μία εγγραφή ή μια συγκεκριμένη καταχώρηση στη βάση δεδομένων. Οι βάσεις δεδομένων επιτρέπουν την αποθήκευση, την τροποποίηση, την ανάκτηση και την διαγραφή δεδομένων. Τέλος, επιτρέπουν επίσης και την εκτέλεση ερωτημάτων (queries) για την αναζήτηση και την ανάκτηση συγκεκριμένων πληροφοριών και όπως έχει προαναφερθεί με συγκεκριμένα κριτήρια. Παρακάτω υλοποιείται ένα παράδειγμα ερωτήσεων κορυφογραμμών (Skyline queries) βασισμένο σε αυτό όπου προαναφέρθηκε προηγουμένως, καθώς και γραμμένο στην γλώσσα προγραμματισμού SQL (Structured Query Language) εφόσον είναι ένα πρότυπο για την επικοινωνία με τις βάσεις δεδομένων. Πιο πρακτικά:

Skyline Query

```
1.  SELECT *
2.  FROM hotels
3.  WHERE price <= X
4.      AND distance <= Y
5.      AND NOT EXISTS (
6.          SELECT *
7.          FROM hotels AS h
8.          WHERE h.price <= X
9.              AND h.distance <= Y
10.             AND (h.price < price OR h.distance < distance)
        )
```

Αλγόριθμος 1: Αλγόριθμος SQL για τη δημιουργία ερώτηση κορυφογραμμής

Η παραπάνω απεικόνιση αποτελεί ένα παράδειγμα ερωτήσεων κορυφογραμμής σε γλώσσα sql. Ακολουθεί αναλυτική περιγραφή του κώδικα γραμμή, γραμμή για την κατανόηση του.

- Στην πρώτη γραμμή υπάρχει η εντολή **SELECT *.** Αυτό το μέρος της εντολής δηλώνει ότι θέλουμε να επιστραφούν όλα τα πεδία(στήλες) από τον πίνακά hotels.
- Στην δεύτερη γραμμή η εντολή **FROM hotels** αναφέρει τον πίνακα hotels από τον οποίο και θα ανακτηθούν τα δεδομένα.
- **WHERE price <= X AND distance <= Y** . Η τρίτη και η τέταρτη γραμμή είναι συνθήκη η οποία ορίζει τα κριτήρια που πρέπει να πληρούνται για να επιλεγούν οι εγγραφές των ξενοδοχείων. Απαιτείται η τιμή της τιμής (price) να είναι μικρότερη ή ίση με τη μεταβλητή X, και η απόσταση(distance) να είναι μικρότερη ή ίση με τη μεταβλητή Y.
- **AND NOT EXISTS (…).** Η πέμπτη εντολή χρησιμοποιεί μία υπό ερώτηση (subquery) με την χρήση της "NOT EXISTS" για να ελέγξει αν δεν υπάρχει άλλη εγγραφή στον πίνακα "hotels" που να είναι καλύτερη σε κάποια από τις διαστάσεις τιμής και απόστασης, από την τρέχουσα επιλεγμένη εγγραφή.
- **SELECT *.** Στην έκτη εντολή όπως και προηγουμένως, χρησιμοποιήθηκε για να δηλώσει ότι θέλει να επιστραφούν, όλα τα πεδία (στήλες) από τον πίνακα "hotels".
- **FROM hotels AS h.** Η έβδομη εντολή αναφέρει στον πίνακα "hotels" και τον αναθέτει το ψευδώνυμο "h". Το ψευδώνυμο μπορεί να χρησιμοποιηθεί για να αναφερθούμε στον πίνακα "hotels" με μια συντόμευση στις υπόλοιπες γραμμές του ερωτήματος.
- **WHERE h.price <= X AND h.distance <= Y.** Εδώ όπως και προηγουμένως στην γραμμή οχτώ και εννιά υλοποιείται μία εντολή-συνθήκη, η οποία ορίζει τα κριτήρια που πρέπει να πληρούνται για να επιλεγούν οι εγγραφές των ξενοδοχείων. Απαιτείται η τιμή της τιμής

(price) να είναι μικρότερη ή ίση με τη μεταβλητή "X" και η απόσταση (distance) να είναι μικρότερη ή ίση με τη μεταβλητή "Y".

- **AND (h.price < price OR h.distance < distance).** Η δέκατη και τελευταία γραμμή ορίζει ένα φίλτρο που ελέγχει αν υπάρχει έστω και μία εγγραφή στον πίνακα "hotels" με μικρότερη τιμή "price" ή μικρότερη απόσταση "distance" από την τρέχουσα εγγραφή του αρχικού κώδικα. Αν υπάρχει τέτοια εγγραφή, τότε η συνθήκη θα είναι αληθής και το NOT EXISTS θα επιστρέψει false για την τρέχουσα εγγραφή, αποκλείοντάς την από το σύνολο κορυφογραμμών.

Τέλος να σημειωθεί ότι ο χρήστης μπορεί να θέσει και παραλλαγές των ερωτήσεων κορυφογραμμών όπου η κάθε μία ερώτηση έχει τον δικό της σκοπό και χρήση. Αυτές οι παραλλαγές των ερωτήσεων κορυφογραμμής αντιμετωπίζουν διάφορες αναλυτικές ανάγκες και έχουν εφαρμογές σε διάφορους τομείς, συμπεριλαμβανομένων των βάσεων δεδομένων, της εξόρυξης δεδομένων και των συστημάτων υποστήριξης αποφάσεων. Σημειώνονται παρακάτω οι εξής:

- Ερωτήσεις Top-k Skyline (Top-k Skyline Queries): Σε μια ερώτηση top-k skyline, ο στόχος είναι η ανάκτηση των k κορυφαίων σημείων skyline με βάση ένα συγκεκριμένο κριτήριο κατάταξης. Αντί να ανακτήσετε όλα τα σημεία skyline, αυτή η παραλλαγή επικεντρώνεται στα k πιο σημαντικά ή ενδιαφέροντα σημεία skyline. (Wu & Chen, 2005)

- Δυναμικές Ερωτήσεις Skyline (Dynamic Skyline Queries): Οι δυναμικές ερωτήσεις skyline λαμβάνουν υπόψη τις αλλαγές στο σύνολο δεδομένων με την πάροδο του χρόνου. Αυτές οι ερωτήσεις χρησιμοποιούνται για την εύρεση σημείων skyline που παραμένουν έγκυρα ακόμη και όταν προστίθενται νέα σημεία δεδομένων ή ενημερώνονται ή διαγράφονται υπάρχοντα. (Zheng, Lu, Wang, & Zhou, 2018)

- Ερωτήσεις Skyline Ομάδων (Group Skyline Queries): Σε μια ερώτηση skyline ομάδων, ο στόχος είναι η εύρεση ομάδων σημείων δεδομένων που αποτελούν skyline ως προς ένα συγκεκριμένο σύνολο διαστάσεων. Αυτή η παραλλαγή είναι χρήσιμη όταν θέλετε να εντοπίσετε ομάδες αμοιβαία μη κυρίαρχων σημείων δεδομένων. (Lai, Lee, & Yin, 2012)

- Αναστροφές Ερωτήσεις Skyline (Reverse Skyline Queries): Οι αναστροφές ερωτήσεις skyline αφορούν την εύρεση σημείων δεδομένων που κυριαρχούν ένα δεδομένο σημείο ερώτησης. Αντί να αναζητήσετε σημεία skyline, ψάχνετε για σημεία δεδομένων που καθιστούν το σημείο ερώτησης ένα μη-σημείο skyline. (Borzsony, Kossmann, & Stocker, 2005)

2.3 Divide and Conquer

Το παράδειγμα διαίρει και βασίλευε χρησιμοποιείται συχνά για την εύρεση της βέλτιστης λύσης ενός προβλήματος. Η βασική του ιδέα είναι να αναλύσει ένα δεδομένο πρόβλημα σε δύο ή περισσότερα παρόμοια, ενδεχομένως πιο απλά υπό προβλήματα, εν συνεχεία να τα λύσει με τη σειρά τους απομονωμένα, για να επιλύσει το δεδομένο αρχικό πρόβλημα. Προβλήματα τα οποία είναι απλά επιλύονται άμεσα και γρήγορα. Για παράδειγμα, για να ταξινομήσει μια δεδομένη λίστα με n φυσικούς αριθμούς, την χωρίζει σε δύο λίστες με περίπου n/2 αριθμούς όπου η καθεμία, ταξινομεί τους αριθμούς με τη σειρά και επιστρέφει δύο αποτελέσματα κατάλληλα για να λάβει την ταξινομημένη έκδοση της δεδομένης λίστας. Αυτή η προσέγγιση είναι γνωστή ως αλγόριθμος ταξινόμησης συγχώνευσης.

Το όνομα "διαίρει και βασίλευε" χρησιμοποιείται μερικές φορές σε αλγόριθμους που μειώνουν κάθε πρόβλημα σε ένα μόνο υπό πρόβλημα, όπως ο αλγόριθμος δυαδικής αναζήτησης για την εύρεση μιας εγγραφής σε μια ταξινομημένη λίστα. Αυτοί οι αλγόριθμοι μπορούν να εφαρμοστούν πιο αποτελεσματικά από τους γενικούς αλγόριθμους διαίρει και βασίλευε. Συγκεκριμένα, εάν χρησιμοποιούν αναδρομή ουράς, μπορούν να μετατραπούν σε απλοί βρόχοι. Σύμφωνα με αυτόν τον ευρύ ορισμό, ωστόσο, κάθε αλγόριθμος που χρησιμοποιεί αναδρομή ή βρόχους θα μπορούσε να θεωρηθεί ως "αλγόριθμος διαίρει και βασίλευε".

Επειδή το διαίρει και βασίλευε επιλύει υπό προβλήματα αναδρομικά, κάθε υπό πρόβλημα πρέπει να είναι μικρότερο από το αρχικό πρόβλημα και πρέπει να υπάρχει μια βασική περίπτωση για τα υπό προβλήματα. Έτσι καλείτε να σκεφτεί κάποιος για τον αλγόριθμο διαίρει και βασίλευε σαν να έχει τρία μέρη:

- Ο αλγόριθμος διαχωρίζει το πρόβλημα σε έναν αριθμό υπό προβλημάτων που είναι σε μικρότερες περιπτώσεις του ίδιου προβλήματος.
- Κάνει την κατάκτηση του στα υπό προβλήματα αυτά λύνοντάς τα αναδρομικά. Εάν είναι αρκετά μικρά, τα λύνει ως βασικές περιπτώσεις.
- Συνδυάζει τις λύσεις των υπό προβλημάτων στη λύση του αρχικού προβλήματος.

Συνοψίζοντας μιλάμε για έναν αλγόριθμο μέσα από τον οποίο προκύπτει μια ισχυρή και εύχρηστη μέθοδος σχεδιασμού αλγορίθμων, ο οποίος έχει ένα μεγάλο πλήθος εφαρμογών. Συνήθως υλοποιείται με αναδρομή αλλά όχι απαραίτητα και χρειάζεται μεγάλη προσοχή στο μέγεθος και στο πλήθος του.

2.4 Naive Skyline

Στην προκειμένη περίπτωση ο υπάρχει και ο αλγόριθμος Naïve Skyline έχοντας διαφορετικό τρόπο επίλυσης και εύρεσης κορυφογραμμής. Ο αλγόριθμος προκειμένου να φτάσει στην εύρεση του συνόλου των σημείων εκείνων που δεν κυριαρχούνται πράττει την αναζήτηση του σειριακά, μέσα από την οποία διαδικασία πραγματοποιείται σύγκριση πάνω σε δυάδες στοιχείων και εύκολα καταλήγει στα σημεία κορυφογραμμών. Παρακάτω φαίνεται παράδειγμα μια αφελής κορυφογραμμής σε ψευδοκώδικα και επεξήγηση.

Naive Skyline

Input: Όρισε **X** τα σημεία της λίστας

Output: Όρισε **S** το σύνολο με τα σημεία κορυφογραμμής

```
1.  Αρχικοποίησε  το S άδειο
2.  Για κάθε Xi από X
3.      κυριαρχείται = ψευδής;
4.      Για κάθε Xj από X
5.          Αν( i == j )
6.              συνέχισε;
7.          Αν(  Xj -> Xi )
8.              κυριαρχείται = αληθής;
9.              σπάσε;
10.     Τέλος Για
11.     Αν( κυριαρχείται == ψευδής)
12.         πρόσθεσε Xi σε S;
13. Τέλος Για
14. Επέστρεψε S;
```

Αλγόριθμος 2:Naive Skyline

Μέσα στο ψευδοκώδικα φαίνεται η αφελής προσέγγιση του Skyline αλγόριθμου. Αρχικά ορίζει Χ τα σημεία της λίστας και κατ' επέκταση S η λίστα η οποία περιλαμβάνει τα σημεία κορυφογραμμών. Στην γραμμή 2 υπάρχει ο βρόγχος επανάληψης Για κάθε Xi που ανήκει στο Χ, ορίζει την μεταβλητή κυριαρχείται ψευδής και έπειτα μετά από επίσης βρόγχο επανάληψης στην γραμμή 4 συγκρίνει όλα τα Xi και τα Xj μεταξύ τους. Αν το κάθε Xj που συγκρίνει με το Xi είναι ίδιο τότε συνεχίζει ο αλγόριθμος την προσπέλαση μέχρι να αλλάξει η μεταβλητή κυριαρχείται σε αληθής και να τρέξει την εντολή σπάσε ή μέχρι να τελειώσει η εσωτερική δομή επανάληψης. Στην γραμμή 11 αν μετά την όλη αναζήτηση η μεταβλητή μείνει ως έχει, τότε προσθέτει το Xi στο σύνολο των κορυφογραμμών S και τέλος επιστρέφει αυτό το σύνολο.

2.5 Block Nested Loop

Θεωρητικό υπόβαθρο

Μια απλή προσέγγιση για τον υπολογισμό του Ορίζοντα των Αποτελεσμάτων είναι η σύγκριση κάθε σημείου α, με κάθε άλλο σημείο β, όπου το α ανήκει στον ορίζοντα αν δεν είναι πρωταρχικό. Ο Αλγόριθμος Block Nested Loop (BNL) βασίζεται σε αυτήν την ιδέα πραγματοποιώντας μια προσπέλαση στο αρχείο δεδομένων και αποθηκεύοντας σε μια λίστα τα ενδεχόμενα σημεία ορίζοντα. Αρχικά, η λίστα περιλαμβάνει το πρώτο σημείο του αρχείου που εξετάζεται από τον αλγόριθμο, ενώ για κάθε επόμενο σημείο α, υπάρχουν τρεις περιπτώσεις:

1. αν το στοιχείο α κυριαρχείται από οποιοδήποτε άλλο στοιχείο της λίστας, απορρίπτεται και δεν περιλαμβάνεται πλέον στο σύνολο του ορίζοντα
2. εάν το στοιχείο α υπερισχύει σε οποιοδήποτε άλλο σημείο της λίστας, τότε όλα αυτά τα στοιχεία που έχει υπερισχύσει το α απορρίπτονται.
3. αν το α δεν ανήκει σε καμία από τις δύο περιπτώσεις, ούτε υπερισχύει, ούτε υπερισχύεται από οποιοδήποτε στοιχείο της λίστας, απλά εισάγεται στο σύνολο του ορίζοντα χωρίς να απορρίπτει κάποιο άλλο στοιχείο.

Η κατάταξη της λίστας της κορυφογραμμής είναι αυτό-οργανωμένη διότι κάθε στοιχείο που περιέχεται σε αυτή κυριαρχεί σε άλλα στοιχεία και μετακινείται στην κορυφή. Αυτό το γεγονός μειώνει τον αριθμό των συγκρίσεων μεταξύ των στοιχείων που υπερισχύουν από πολλά άλλα στοιχεία στη λίστα. Ένα πρόβλημα του Block Nested Loop αλγορίθμου, είναι ότι η λίστα μπορεί να γίνει μεγαλύτερη από την κύρια μνήμη. Όταν αυτό συμβαίνει, όλα τα στοιχεία που ανήκουν στην τρίτη περίπτωση (οι περιπτώσεις (1) και (2) δεν αυξάνουν το μέγεθος της λίστας) προστίθενται σε ένα προσωρινό αρχείο. Αυτό απαιτεί πολλαπλές επαναλήψεις του αλγορίθμου. Συγκεκριμένα, μετά την ολοκλήρωση της προσπέλασης του αρχείου δεδομένων, όλα τα στοιχεία που έχουν εισαχθεί στη λίστα πριν από τη δημιουργία του προσωρινού αρχείου είναι εξασφαλισμένα να είναι στο σύνολο της κορυφογραμμής και θα εξαχθούν ως αποτελέσματα. Τα υπόλοιπα στοιχεία πρέπει να συγκριθούν με αυτά του προσωρινού αρχείου και ο αλγόριθμος θα εκτελέσει ξανά την διαδικασία χρησιμοποιώντας αυτή την φορά ως είσοδο το προσωρινό αρχείο και όχι την αρχική λίστα.

Το πλεονέκτημα στην ιδιότητα του Block Nested Loop είναι η ευρεία εφαρμογή του, καθώς μπορεί να χρησιμοποιηθεί για οποιαδήποτε διάσταση χωρίς την ανάγκη ευρετηρίασης ή ταξινόμησης του αρχείου δεδομένων. Ωστόσο, υπάρχουν ορισμένα βασικά προβλήματα που αντιμετωπίζει. Πρώτον, εξαρτάται από την κύρια μνήμη, και μια μικρή μνήμη μπορεί να οδηγήσει σε πολλαπλές επαναλήψεις του αλγορίθμου. Δεύτερον, ο BNL δεν είναι κατάλληλος για προοδευτική επεξεργασία, καθώς πρέπει να διαβάσει ολόκληρο το αρχείο δεδομένων πριν επιστρέψει το πρώτο σημείο του ορίζοντα.

Για την αντιμετώπιση αυτών των προβλημάτων, υπάρχουν αλγόριθμοι όπως ο "sort first skyline" (SFS) που βασίζονται στο BNL και αντιμετωπίζουν αυτά τα προβλήματα. Ο αλγόριθμος SFS πρώτα ταξινομεί ολόκληρο το σύνολο δεδομένων βάσει μιας (μονότονης) συνάρτησης προτίμησης. Στη συνέχεια, τα στοιχεία εισάγονται στη λίστα με αύξουσα σειρά των βαθμολογιών τους, γιατί οι βαθμοί με χαμηλότερη βαθμολογία είναι πιθανό να κυριαρχούν σε μεγάλο αριθμό σημείων, καθιστώντας έτσι το κλάδεμα των δεδομένων της λίστας πιο αποτελεσματικό.

Υλοποίηση

Ο αλγόριθμος BNL ξεκινά με μια λίστα L όλων των σημείων δεδομένων και μια κενή λίστα σημείων οριζοντιογραφίας. Ο αλγόριθμος ελέγχει πρώτα αν η L είναι κενή ή όχι- αν συμβαίνει αυτό, ο αλγόριθμος τερματίζει αμέσως. Διαφορετικά, εισάγουμε το πρώτο σημείο δεδομένων της L στο S (υποψήφιο σημείο ορίζοντα). Στη συνέχεια, ο αλγόριθμος διατρέχει με βρόχο όλα τα εναπομείναντα σημεία δεδομένων του L και εκτελεί ορισμένες πράξεις για την ενημέρωση του περιεχομένου του καταλόγου S. Συγκεκριμένα, εάν ένα σημείο p του L κυριαρχεί σε ορισμένα σημεία του S, προστίθεται στο S και όλα τα σημεία που κυριαρχούνται από το p αφαιρούνται από το S. Στην επόμενη περίπτωση, το σημείο p δεν κυριαρχείται ή δεν κυριαρχεί σε ορισμένα σημεία του S, θεωρείται υποψήφιο σημείο οριζόντιου ορίζοντα και προστίθεται στο S. Αυτή είναι μια αφελής προσέγγιση του αλγορίθμου BNL για τον υπολογισμό των σημείων οριζόντιου ορίζοντα και είναι αποτελεσματική για ορισμένους μικρούς και χαμηλής διάστασης χώρους δεδομένων.

```
BNL algorithm
Input: L – the list of all data points
Skyline S = ∅

// If the list L is empty, terminate the algorithm
IF L is empty
   Terminates the algorithm, returns S with no element

Insert the first point in L into S

FOR each point p in L
    IF dominatesInSet(p, S)
        Add p to S
        Remove all points in S that are dominated by p
    ELSE IF !(dominatesInSet(p, S) || dominatedInSet(p, S))
        Add p to S
END FOR

// Return the skyline points
RETURN S
End BNL algorithm
```

Αλγόριθμος 3: BNL (Block Nested Loop)

2.6 Branch and Bound Skyline (BBS) Algorithm

Θεωρία

Στην πραγματικότητα, μπορεί να εφαρμοστεί οποιαδήποτε μορφή δείκτη για να διαχωρίσει τον χώρο των δεδομένων μας, αλλά στην υλοποίηση για το BBS θα επιλεχθεί το R-Tree (Guttman, 1984) λόγω της απλότητας και της αποδοτικότητάς του. Ο αλγόριθμος BBS βασίζεται στο παράδειγμα branch-and-bound. Αρχικά, ο αλγόριθμος ξεκινά από τη ρίζα του R-Tree και εισάγει όλες τις καταχωρήσεις της ρίζας σε μια δομή δεδομένων σωρού (στην περίπτωσή αυτή χρησιμοποιείτε μια ουρά προτεραιότητας) ταξινομημένες ανάλογα με τις τιμές mindist (το άθροισμα όλων των διαστάσεων μιας καταχώρησης). Στη συνέχεια, ο αλγόριθμος εξάγει την κορυφαία καταχώρηση του σωρού με την ελάχιστη τιμή mindist και την επεκτείνει (μπορεί απλά να αντλήσει το πρώτο στοιχείο της ουράς προτεραιότητας στην υλοποίησή). Η διαδικασία επέκτασης αφαιρεί την εν λόγω καταχώρηση από τον σωρό και εισάγει όλα τα παιδιά της μέσα. Ο αλγόριθμος, προχωρά με την επόμενη καταχώρηση σωρού που έχει την ελάχιστη τιμή mindist. Κάθε φορά που αντλεί μια καταχώρηση p από τη σωρό, ελέγχει την υπεροχή της έναντι όλων των καταχωρήσεων στη λίστα κορυφογραμμών. Εάν η κορυφαία καταχώρηση δεν υποκαθίσταται από καμία εγγραφή στην λίστα κορυφογραμμών και η p αποτελεί μια ενδιάμεση καταχώρηση, τότε λαμβάνει όλα τα παιδιά της και επανεξετάζει την υπεροχή τους έναντι των εγγραφών στην λίστα κορυφογραμμών. Αν ένα παιδί της p δεν υποκαθίσταται από καμία εγγραφή στην λίστα κορυφογραμμών, τότε προστίθεται στον σωρό. Αν η καταχώρηση που αντλήθηκε από τον σωρό είναι ένα σημείο δεδομένων, τότε προστίθεται στην λίστα κορυφογραμμών. Ο αλγόριθμος ολοκληρώνεται όταν η σωρός είναι κενή.

Υλοποίηση

Αρχικά ο αλγόριθμός ενεργοποιεί την ενσωματωμένη λειτουργία R-Tree, με σκοπό την εύρεση των καταχωρήσεων που ανήκουν στη ρίζα του δέντρου R-Tree, και τις τοποθετεί σε έναν σωρό (μια ουρά προτεραιότητας στην παρούσα υλοποίηση). Με αυτόν τον τρόπο, προετοιμάζει τον σωρό με αρχικές υποψήφιες καταχωρήσεις. Στο επόμενο στάδιο, ο αλγόριθμος επαναλαμβάνει μια διαδικασία για κάθε στοιχείο στον σωρό, με στόχο τον υπολογισμό των ορίων και την ενημέρωση του περιεχομένου του σωρού. Αναλυτικά, σε κάθε επανάληψη, αφαιρεί το πρώτο στοιχείο από την κορυφή του σωρού και ελέγχει τη συνθήκη κυριαρχίας με όλα τα σημεία στη λίστα skyline. Αν η κορυφή κυριαρχείται από οποιοδήποτε σημείο στην κατάλληλη λίστα skyline, τότε το απορρίπτει αμέσως χωρίς περαιτέρω εξέταση. Διαφορετικά, εάν η κορυφή είναι μια ενδιάμεση καταχώρηση, την επεκτείνει και εξετάζει τη συνθήκη κυριαρχίας στα παιδιά της. Καθώς οι καταχωρήσεις στον σωρό ταξινομούνται με βάση την τιμή mindist, ο αλγόριθμος επισκέπτεται τις καταχωρήσεις του δέντρου με αύξουσα σειρά βάσει του mindist. Επιπλέον, κάθε σημείο δεδομένων θα εξεταστεί, εκτός εάν ένας από τους προγόνους του έχει απορριφθεί. Στην πραγματικότητα, ο αλγόριθμος προσπελάζει αμέσως τα σημεία του ορίζοντα χωρίς να χρειαστεί να προσπελάσει μεγάλο μέρος του δέντρου.

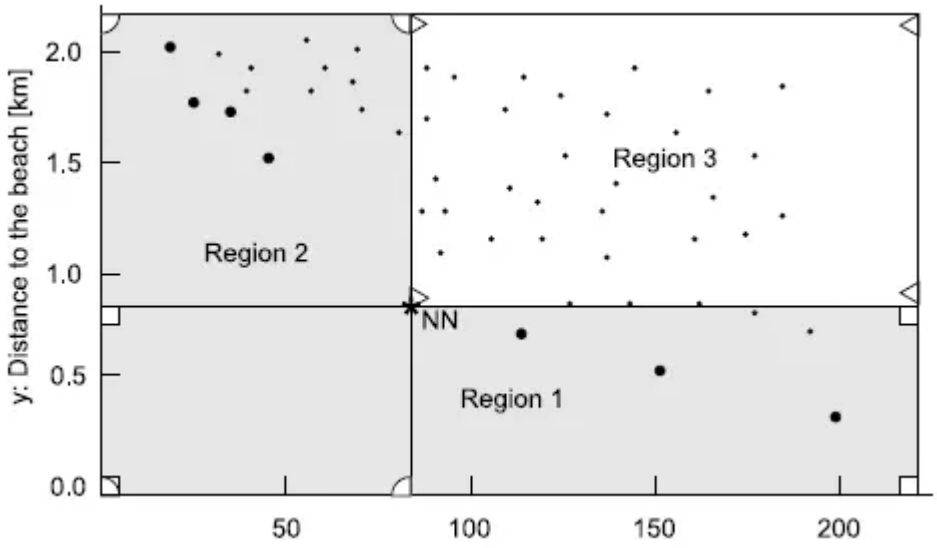
```
BBS algorithm
Input: Dataset D (indexed with R-Tree R)
Skyline S = ∅
Heap H = ∅

// If there is no data points, terminate the algorithm
IF D is empty
   Terminates the algorithm, returns S with no element
ELSE
   Add all entries of the root node of R-Tree in the heap H

   // Repeat the whole process while the heap is not empty
   WHILE (H ≠ ∅)
     Get the first element of the heap (head) – e
        IF e is dominated by some point in S
           Discard e
        ELSE
           IF e is an intermediate entry
              FOR each child q of e
                 IF q is not dominated by some point in S
                    Insert q into H
                 ELSE
                    Insert e into S

// Return the skyline points
RETURN S
```

Αλγόριθμος 4: (BBS) Branch and Bound Skyline Algorithm

2.7 Nearest Neighbor (NN) Algorithm

Θεωρία

Ο αλγόριθμος Nearest Neighbor (NN) (Kossmann, Ramsak, & Rost, 2002) έχει ως βάση την λειτουργία Nearest Neighbor Search στο R-Tree. Αυτό που κάνει είναι ότι διαιρεί τον χώρο δεδομένων αναδρομικά μετά την ανακάλυψη ενός πλησιέστερου γείτονα και εφαρμόζει τη λειτουργία αναζήτησης σε υπό-χώρους. Πιο συγκεκριμένα, ο αλγόριθμος εκτελεί τη λειτουργία αναζήτησης του πλησιέστερου γείτονα (Nearest Neighbor Search) για να εντοπίσει το σημείο που βρίσκεται στην ελάχιστη απόσταση από την αρχή των αξόνων (σημείο αρχής). Αυτό το σημείο αποτελεί αναπόφευκτα ένα σημείο κορυφογραμμής. Μετά την εντοπισμό της πρώτης κορυφογραμμής (skyline), ο χώρος των δεδομένων διαιρείται σε τρεις περιοχές (στον 2D χώρο).



Εικόνα 2: Διαίρεση χώρου μετά την ανακάλυψη του πρώτου πλησιέστερου γείτονα

Όλα τα σημεία που βρίσκονται στην περιοχή 3 μπορούν να αποκλειστούν στην περαιτέρω διαδικασία, καθώς αυτά τα σημεία είναι κυριαρχούμενα από τον πρώτο πλησιέστερο γείτονα (NN στο σχήμα). Οι περιοχές 1 και 2 θα προστεθούν σε μια λίστα καθηκόντων, διότι ενδέχεται να περιέχουν άλλα σημεία του ορίζοντα. Ο πλησιέστερος γείτονας (NN) θα αξιολογήσει τη λίστα καθηκόντων μέχρι να αδειάσει, και τότε ο αλγόριθμος θα ολοκληρωθεί. Όσο η λίστα καθηκόντων δεν είναι άδεια, ο πλησιέστερος γείτονας θα αφαιρέσει μια περιοχή από τη λίστα και θα επαναλάβει αναδρομικά την ίδια διαδικασία.

Η λειτουργία Nearest Neighbor Search είναι αποδοτική όταν τα δεδομένα είναι ευρετήριο με R-Tree, καθώς ο πλησιέστερος γείτονας (NN) μπορεί να επιστρέψει αρχικά κάποια αποτελέσματα. Ωστόσο, όσο

αυξάνεται ο αριθμός των διαστάσεων του χώρου δεδομένων, η απόδοση της Nearest Neighbor Search μειώνεται σημαντικά, καθώς η λειτουργία αυτή γίνεται ακριβή.

Ο αλγόριθμος NN δεν είναι εφαρμόσιμος εάν το ερώτημα κορυφογραμμής περιλαμβάνει μια διάσταση που δεν είναι ευρετηριασμένη, δηλαδή αν δεν υπάρχει δυνατότητα αναζήτησης σε αυτή την διάσταση. Επίσης, εάν το ερώτημα της κορυφογραμμής περιλαμβάνει μεγάλες ενώσεις ή απαιτεί λειτουργία group-by, τότε ο αλγόριθμος NN δεν είναι κατάλληλος.

Σε αυτές τις περιπτώσεις, η απόδοση του αλγορίθμου μπορεί να μειωθεί σημαντικά και ενδέχεται να απαιτηθούν εναλλακτικές μέθοδοι αναζήτησης δεδομένων, για να επιτευχθεί η επιθυμητή λειτουργικότητα.

Υλοποίηση

Η boundedNNSearch είναι η καρδιά του αλγορίθμου και αναλαμβάνει την αναζήτηση περιορισμένου κοντινότερου γείτονα. Αυτή η λειτουργία δέχεται ως είσοδο την αρχή των αξόνων, την κατασκευή ενός ορθογωνίου από την αρχή και το σημείο HEAD (x, y) (με H(x) να είναι το πλάτος του ορθογωνίου και H(y) το ύψος του ορθογωνίου) και τον αριθμό των σημείων που πρέπει να βρεθούν. Στην περίπτωσή σας, η boundedNNSearch έχει ως αρμοδιότητα να εντοπίσει τον πρώτο κοντινότερο γείτονα στην αρχή, με την προϋπόθεση ότι το σημείο αυτό βρίσκεται αυστηρά εντός του ορθογωνίου που ορίζεται παραπάνω.

Ο αλγόριθμος NN είναι αποτελεσματικός για την ανάκτηση μιας πρώτης ομάδας σημείων που ανήκουν στον ορίζοντα, λόγω της γρήγορης λειτουργίας αναζήτησης του πλησιέστερου γείτονα στο R-Tree. Ωστόσο, όταν απαιτείται η ανάκτηση όλων των σημείων skyline σε έναν μεγάλο και πολυδιάστατο χώρο δεδομένων, η απόδοση του αλγορίθμου NN μειώνεται σημαντικά. Αυτό συμβαίνει εξαιτίας των περιττών ερωτημάτων που δεν επιστρέφουν αποτελέσματα που πρέπει να εκτελεί ο αλγόριθμος NN. Ο αλγόριθμος NN τερματίζει όταν το ερώτημα του πλησιέστερου γείτονα δεν επιστρέφει κανένα σημείο εντός του τρέχοντος χώρου δεδομένων (κενό ερώτημα).

Ωστόσο, σε πολλές περιπτώσεις, ο αλγόριθμος NN ανακτά σημεία του ορίζοντα που έχουν ήδη εντοπιστεί σε προηγούμενα ερωτήματα. Αυτές οι επιπλέον ανακτήσεις προκαλούν την πολλαπλή προσπέλαση ενός κόμβου στο δέντρο και επηρεάζουν αρνητικά την απόδοση του αλγορίθμου. Ένα άλλο πρόβλημα του NN είναι ο μεγάλος αριθμός των εργασιών που πρέπει να εκτελέσει, ο οποίος μπορεί να υπερβαίνει το μέγεθος του χώρου δεδομένων, ακόμη και αν δεν λάβουμε υπόψη τις περιττές αναζητήσεις.

NN algorithm

Input: Dataset D (indexed with R-Tree R)

To-do list T = {(∞, ∞)}

Skyline S = ∅

// If there is no data points, terminate the algorithm

IF D is empty

  Terminates the algorithm, returns S with no element

ELSE

  // Find the first nearest neighbor to the origin

  P (x, y) = nearestNeighborSearch (R, origin, 1)

  // Add the first nearest neighbor to the skyline list

  Add P (x, y) to S

  // Add two points after discover the first nearest neighbor to the

  // to-do list. Points in the to-do list will be used to construct the

  // space (rectangle in 2D space) later.

  Add P-a (P(x), MAX_COORDINATE_VALUE) to T

  Add P-b (MAX_COORDINATE_VALUE, P(y)) to T

// Repeat the whole process while the to-do list is not empty

WHILE (T ≠ ∅)

  // Fetch the first element from the to-do list

  HEAD (x, y) = get the first element from T

  // Perform the bounded nearest neighbor search

  SKYLINE (x, y) = boundedNNSearch (origin, Rec(origin, HEAD (x, y)), 1)

  // Add the result of bounded nearest neighbor search and

  // divided spaces to the to-do list (if they exist)

  IF SKYLINE (x, y) exists

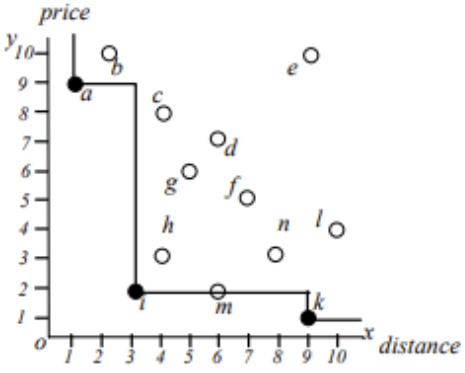
    Add SKYLINE (x, y) to S

```
    Add (SKYLINE(x), HEAD(y)) to T
    Add (HEAD(y), SKYLINE(x)) to T

// Return the skyline points
RETURN S
End NN algorithm
```

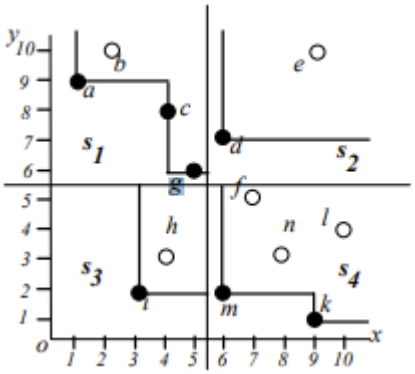
Αλγόριθμος 5:(NN) Nearest Neighbor Algorithm

2.8 *Bitmap Technique (Τεχνική Δυαδικής Προσέγγισης)*

Το bitmap είναι ένας τρόπος κωδικοποίησης και αποθήκευσης εικόνων, γραφικών και άλλων στοιχείων σε ψηφιακή μορφή. Συγκεκριμένα αυτή η τεχνική κωδικοποιεί σε δυαδικά αρχεία όλες τις πληροφορίες από τα αντικείμενα που απαιτούνται για να αποφασιστεί εάν ένα σημείο βρίσκεται στη κορυφογραμμή. Πιο συγκεκριμένα, ένα δεδομένο σημείο p = (p1, p2, ..., pd), όπου d είναι ο αριθμός των διαστάσεων, αντιστοιχίζεται σε ένα m-δυαδικό διάνυσμα, όπου m είναι ο συνολικός αριθμός διακριτών τιμών σε όλες τις διαστάσεις. Έστω ki ο συνολικός αριθμός διακριτών τιμών στην i-οστή διάσταση (δηλαδή m = ∑i=1~dki). Στο Σχήμα 1, για παράδειγμα, υπάρχουν k1 = k2 = 10 διακριτές τιμές στις διαστάσεις x και y, καθώς και m = 20. Ας γίνει υπόθεση ότι pi είναι ο ji-οστός μικρότερος αριθμός στον i-οστό άξονα· τότε αναπαρίσταται από ki δυαδικά ψηφία, όπου τα (ki − ji +1) πιο σημαντικά ψηφία είναι 1 και τα υπόλοιπα είναι 0. Ο πίνακας 2.1 δείχνει τα δυαδικά αρχεία για τα σημεία στο Σχήμα 1.1. Αφού το σημείο α έχει τη μικρότερη τιμή (1) στον άξονα x, όλα τα δυαδικά ψηφία του a1 είναι 1.



Εικόνα 4: Γράφημα για την ένδειξη τελικής κορυφογραμμής



Εικόνα 3: Γράφημα για την ένδειξη των τεσσάρων κορυφογραμμών

Στις παραπάνω απεικονίσεις φαίνονται δύο σχεδιαγράμματα, τα οποία προκύπτουν από ένα απλό παράδειγμα και θα χρησιμοποιηθούν για την περεταίρω κατανόηση της τεχνικής Bitmap. Στο πρώτο

σχεδιάγραμμά υπάρχουν αντικείμενα-σημεία τα οποία έχουν το καθένα συγκεκριμένες συντεταγμένες και λαμβάνουν την ανάλογη θέση μέσα στο τεταρτημόριο του σχεδιαγράμματος στις διαστάσεις χ και y. Επίσης φαίνεται χαρακτηριστικά και η έντονη μαύρη γραμμή η οποία είναι η κορυφογραμμή, και φανερώνει τα σημεία τα οποία βρίσκονται μέσα στο σύνολο της. Το πως προκύπτει αυτή κορυφογραμμή, φαίνεται πιο αναλυτικά στο δεύτερο σχέδιο καθώς δείχνει τα 4 στάδια όπου και πέρασε για να μπορέσει να δημιουργήσει την τελική κορυφογραμμή. Πιο αναλυτικά, υπολογίζεται ένα μέρος της κορυφογραμμής των χωρίζοντας το σε τμήματα χρησιμοποιώντας έναν αλγόριθμο μνήμης, κύριας μνήμης (π.χ., [SM88, M91]), και η τελική κορυφογραμμή προκύπτει από τη συγχώνευση των μερικών. Τα δεδομένα για το δεύτερο σχεδιάγραμμα βρίσκονται στο σύνολο των δεδομένων του παρακάτω πίνακα. Ο χώρος των δεδομένων διαιρείται σε 4 τμήματα s1, s2, s3, s4, με κορυφαία σημεία {a, c, g}, {d}, {i}, {m, k}, αντίστοιχα. Για να προκύψει η τελική κορυφογραμμή, πρέπει να αφαιρέσει τα σημεία που κυριαρχούνται από κάποιο άλλο σημείο σε άλλα τμήματα.

Προφανώς, όλα τα σημεία της κορυφογραμμής του s3 πρέπει να εμφανίζονται στην τελική κορυφογραμμή, ενώ αυτά του s2 απορρίπτονται αμέσως επειδή υποκαθίστανται από οποιοδήποτε σημείο στο s3 (στην πραγματικότητα, το s2 πρέπει να ληφθεί υπόψη μόνο αν το s3 είναι άδειο). Κάθε σημείο της κορυφογραμμής στο s1 συγκρίνεται μόνο με τα σημεία στο s3, επειδή κανένα σημείο στο s2 ή s4 δεν μπορεί να υποκαταστήσει αυτά του s1. Σε αυτό το παράδειγμα, τα σημεία c, g αφαιρούνται επειδή υποκαθίστανται από το i. Με ανάλογο τρόπο, η κορυφογραμμή του s4 συγκρίνεται επίσης με τα σημεία του s3, με αποτέλεσμα την αφαίρεση του m. Τελικά, ο αλγόριθμος τερματίζει με τα υπόλοιπα σημεία {a, i, k}.

id	coordinate	bitmap representation
a	(1,9)	(1111111111, 1100000000)
b	(2,10)	(1111111110, 1000000000)
c	**(4,8)**	(1111111000, 1110000000)
d	(6,7)	(1111100000, 1111000000)
e	(9,10)	(1100000000, 1000000000)
f	(7,5)	(1111000000, 1111110000)
g	(5,6)	(1111110000, 1111100000)
h	(4,3)	(1111111000, 1111111100)
i	(3,2)	(1111111100, 1111111110)
k	(9,1)	(1100000000, 1111111111)
l	(10,4)	(1000000000, 1111111000)
m	(6,2)	(1111100000, 1111111110)
n	(8,3)	(1110000000, 1111111100)

Πίνακας 2:Τα στοιχεία όλων των εγγραφών

Ας θεωρήσουμε τώρα ότι θέλουμε να αποφασίσουμε εάν ένα σημείο, για παράδειγμα το c με αναπαράσταση σε bitmap (1111111000, 1110000000), ανήκει στη κορυφογραμμή. Τα πιο σημαντικά ψηφία που έχουν την τιμή 1 είναι το 4ο και το 8ο, στις διαστάσεις x και y, αντίστοιχα. Ο αλγόριθμος δημιουργεί δύο δυαδικές ακολουθίες, cX = 1110000110000 και cY = 0011011111111, ενώνοντας τα αντίστοιχα ψηφία (δηλαδή το 4ο και το 8ο) από κάθε σημείο. Στον Πίνακα 2.1, αυτές οι δυαδικές ακολουθίες (που φαίνονται με έντονη γραφή) περιλαμβάνουν 13 ψηφία (ένα από κάθε αντικείμενο, από το a μέχρι το n). Τα 1 στο αποτέλεσμα της πράξης cX και cY = 0010000110000, υποδεικνύουν τα σημεία που κυριαρχούν στο c,

δηλαδή το c, το h και το i. Προφανώς, αν υπάρχει περισσότερο από ένα 1, το σημείο που εξετάζεται δεν ανήκει στη κορυφογραμμή 2. Οι ίδιες λειτουργίες επαναλαμβάνονται για κάθε σημείο στο σύνολο δεδομένων, για να λάβουμε ολόκληρη τη κορυφογραμμή.

Η αποδοτικότητα του bitmap βασίζεται στην ταχύτητα των δυαδικών λογικών λειτουργιών. Η προσέγγιση αυτή μπορεί να επιστρέψει γρήγορα τα πρώτα λίγα σημεία της κορυφογραμμής σύμφωνα με τη σειρά εισαγωγής τους, για παράδειγμα, αλφαβητική σειρά, αλλά δεν μπορεί να προσαρμοστεί σε διαφορετικές προτιμήσεις των χρηστών, που αποτελεί μια σημαντική ιδιότητα ενός καλού αλγορίθμου κορυφογραμμής. Επιπλέον, ο υπολογισμός ολόκληρης της κορυφογραμμής είναι ακριβός, διότι για κάθε εξεταζόμενο σημείο, πρέπει να ανακτήσει τα δυαδικά αρχεία όλων των σημείων για να προσπελάσει τις αντιστοιχίσεις. Επίσης, η κατανάλωση χώρου μπορεί να είναι απαγορευτική, εάν ο αριθμός των διακριτών τιμών είναι μεγάλος. Τέλος, η τεχνική δεν είναι κατάλληλη για δυναμικά σύνολα δεδομένων όπου οι εισαγωγές μπορεί να αλλάξουν την κατάταξη των τιμών των γνωρισμάτων.

2.9 Sort First Skyline (SFS) Algorithm

Ο αλγόριθμος SFS (Sort First Skyline) (Bartolini, Ciaccia, & Patella, 2008) τον οποίο των προτείνουν οι συγγραφείς του οι Chomicki, Godfrey, Gryz, & Liang (2005) σε δημοσίευση τους, βελτιώνει σημαντικά την απόδοση του αλγορίθμου BNL καθώς και έχουν ομοιότητες. Αυτό συμβαίνει για τον λόγο ότι ο SFS γίνεται ένα με τον BNL και εφόσον ομαλοποιηθούν οι διαστάσεις των σημείων σε τιμές από 0 έως 1, τα σημεία αυτά, τα οποία χρησιμοποιούνται από τον αλγόριθμο ως σύνολο δεδομένων προς είσοδο, με την χρήση της εντροπίας, όπου ουσιαστικά μέσω της μεθοδολογίας αυτής, προστίθενται όλες οι διατάσεις του κάθε σημείου ξεχωριστά μεταξύ τους και μέσα από το τελικό αποτέλεσμα της κάθε σειράς, ο αλγόριθμος τα κατατάσσει κατά αύξουσα σειρά. Με την μεθοδολογία αυτή ο Jan Chomicki πετυχαίνει το εξής: Μεταξύ δύο σημείων το σημείο το οποίο κυριαρχεί θα είναι και αυτό, το οποίο θα ελεγχθεί και πρώτο και πρέπει να σημειωθεί ότι τα ζεύγη των σημείων όπου πρέπει να ελεγχθούν από τον αλγόριθμο με την πάροδο του χρόνου μειώνονται παρουσιάζοντας στην προσπέλαση των σημείων του αλγορίθμου σημαντική πρόοδο.

Η συνεισφορά της εντροπίας στην όλη διεκπεραίωση της λειτουργίας του αλγορίθμου είναι να βγάλει το τελικό αποτέλεσμα του κάθε σημείου από το σύνολο των δεδομένων από το μικρότερο στο μεγαλύτερο. Με βάση αυτό το αποτέλεσμα αυτό, ο αλγόριθμος ταξινομεί το σύνολο των δεδομένων. Όπως προαναφέρθηκε ο αλγόριθμος SFS έχει την ίδια λειτουργία με αυτή του BNL, καθώς κρατάει στον buffer τα σημεία τα οποία βρίσκει και επιλέγει να ξεχωρίσει ώστε να είναι σημεία κορυφογραμμής. Στην αρχή της διαδικασίας, στην κύρια μνήμη, ο buffer είναι κενός και εφόσον εισάγεται ένα στοιχείο σε αυτόν, ο αλγόριθμος δουλεύει με τον τρόπο, ότι πηγαίνει και διαβάζει το επόμενο στοιχείο. Βέβαια ο τρόπος με τον οποίο συγκρίνει τα στοιχεία μεταξύ τους δεν αλλάζει και όπως προαναφέρθηκε όταν ένα σημείο δεν κυριαρχείται από κανένα άλλο σημείο τότε εισάγεται μέσα στον buffer.

Για τον υπολογισμό της εντροπίας σύμφωνα με τους συγγραφείς Chomicki, Godfrey, Gryz, και Liang προτείνεται ο εξής τύπος:

$$Ed(p) = \sum_{i=1}^{d} \quad ln(p'd_i \ + 1)(1)$$

Για την περαιτέρω κατανόηση του μαθηματικού τύπου πρέπει να ειπωθεί ότι όπου p'. di είναι οι τιμές οι ομαλοποιημένες με εύρος από 0 έως 1. Μέσα από τον υπολογισμό της εντροπίας μπορεί και προκύπτει συγκεκριμένο αποτέλεσμα για το κάθε σημείο καθώς μέσα από αυτό ο αλγόριθμος μπορεί και καταλαβαίνει ποιο σημείο θα τεθεί εκτός κορυφογραμμής. Αν το αποτέλεσμα έχει υψηλή τιμή τότε το σημείο απορρίπτεται. Παρακάτω ακολουθεί ψευδό κώδικας του αλγορίθμου.

Sort First Skyline (SFS) algorithm

Input: Ταξινομημένο σύνολο δεδομένων D (Heap).

Output: Τα σημεία κορυφογραμμών από το σύνολο δεδομένων D

```
1.  Unfinished = True
2.  While (unfinished)do
3.    T=open_cursor(Heap)
4.    Unfinished = False
5.    While (next_tuple(T,t)) do
6.      if("T is not dominated") then
7.      if("window is full") then
8.        unfinished = True
9.        break
10.     else
11.       "Add t to window"
12. if (unfinished) then
13.   S = open_new_file_(second_pass)
14.   write(S,t)
15.   while(next_tuple(T,t)) do
16.     if ("t is not domintated") then
17.       Write(S,t)
18. free(Heap)
19. close (S)
20. Heap=SecondPass
21. "Write window tuples to output"
22. "Clear window"
```

Αλγόριθμος 6: Αλγόριθμος Sort First Skyline (SFS)

Παραπάνω φαίνεται ο ψευδοκώδικας του αλγορίθμου SFS. Αρχικά ο αλγόριθμος παίρνει σαν είσοδο του ένα ταξινομημένο σύνολο δεδομένων, στην όλη λειτουργία όμως, σημαντικό ρόλο έχει ο buffer καθώς εάν όλα τα σημεία δεν χωράνε στην μνήμη του, τότε ο αλγόριθμος θα πραγματοποιήσει πολλαπλά περάσματα προκειμένου να ελεγχθούν όλα τα σημεία. Εάν όλα τα σημεία χωράνε στην μνήμη του buffer, τότε ο αλγόριθμος θα χρειαστεί μόνο ένα πέρασμα.

Για να μπορέσουμε να ομαλοποιήσουμε τα σημεία σε τιμές από 0 έως 1, οι συγγραφείς Kalyvas & Tzouramanis προτείνουν την εξής μέθοδο, μέσα από την οποία εντοπίζει την μεγαλύτερη τιμή σε όλο το σύνολο των δεδομένων και έπειτα διαιρεί όλα τα σημεία σε όλες τις διαστάσεις. Η μέθοδος με αυτή την συμπεριφορά είναι ο εξής τύπος:

$$f(p.d_i) = \left(p.\frac{d_i}{max}\right)$$

Παρακάτω ακολουθεί ένα παράδειγμα για την καλύτερη κατανόηση στην λειτουργία του αλγορίθμου, και το παράδειγμα είναι βασισμένο πάνω σε σπίτια και την απόσταση που έχουν από το κέντρο της πόλης. Ας υποθέσουμε λοιπόν ότι έχουμε 11 σπίτια με συγκεκριμένη τιμή και απόσταση.

Σπίτι	**Τιμή**	**Απόσταση**
Σ1	100	1500
Σ2	1400	500
Σ3	700	600
Σ4	1300	1000
Σ5	900	1300
Σ6	1600	100
Σ7	400	300
Σ8	200	1200
Σ9	1000	200
Σ10	500	1400
Σ11	500	900

Πίνακας 3: Στοιχεία των εγγραφών σε σπίτια στο κέντρο της πόλης

Για να μπορέσει να λειτουργήσει ο αλγόριθμος αρχικά χρησιμοποιείτε η συνάρτηση εντροπίας ώστε να μπορέσουν να ομαλοποιηθούν οι τιμές και να ταξινομηθούν κατά αύξουσα σειρά. Αρχικά ο buffer είναι

κενός και ο αλγόριθμος ξεκινάει εισάγοντας ως πρώτο σημείο το Σ7. Έπειτα ως δεύτερο σημείο ακολουθεί το σημείο Σ9 το οποίο όμως δεν μπορεί να συγκριθεί με το Σ7 και ο αλγόριθμος το εισάγει κατευθείαν στην προσωρινή μνήμη. Το σημείο Σ3 βλέπουμε ότι κυριαρχείται από το Σ7 επομένως απορρίπτεται και δεν εισχωρείτε στην προσωρινή μνήμη. Το Σ8 δεν μπορεί να συγκριθεί και αυτό με το Σ7 αλλά και με το Σ9 τα οποία βρίσκονται ήδη μέσα στον buffer και έτσι και αυτό με την σειρά του προστίθεται μέσα στον buffer. Το σημείο Σ11 από την άλλη κυριαρχείται από τον Σ7, οπότε απορρίπτεται. Το σημείο Σ1 κυριαρχείται από τα σημεία τα οποία είναι ήδη μέσα στην μνήμη όπως αντίστοιχα και για το Σ6. Τα υπόλοιπα τέσσερα σημεία κυριαρχούνται και αυτά από κάποιο σημείο το οποίο είναι μέσα στην προσωρινή μνήμη. Είναι αξιοσημείωτο λοιπόν ότι λόγο της σωστής αύξουσας ταξινόμησης, τα πιο κυρίαρχα σημεία είναι και αυτά τα οποία είναι και πρώτα στο σύνολο δεδομένων για επεξεργασία από τον αλγόριθμο. Το αποτέλεσμα αυτό εξασφαλίζει τη μέγιστη απόρριψη σημείων με ελάχιστες συγκρίσεις, δίνοντας έτσι 5 κορυφαία σημεία από τα 11 σημεία του συνόλου, για τον σχηματισμό κορυφογραμμής. Παρακάτω φαίνεται ο πίνακας με τα αποτελέσματα, μετά τον υπολογισμό της εντροπίας και μετά την προσπέλαση του αλγορίθμου.

Σπίτι	Τιμή	Απόσταση	Εντροπία	Σημεία που κυριάρχησε
Σ7	0,04	0,03	0,068779515	6
Σ9	0,1	0,02	0,115112807	2
Σ3	0,07	0,06	0,125927557	-
Σ8	0,02	0,12	0,133131313	2
Σ11	0,05	0,09	0,13496786	-
Σ1	0,01	0,15	0,149712273	0
Σ6	0,16	0,01	0,158370336	0
Σ2	0,14	0,05	0,179818427	-
Σ10	0,05	0,14	0,179818427	-
Σ5	0,09	0,13	0,208395329	-
Σ4	0,13	0,1	0,217527813	-

Πίνακας 4: Οι εγγραφές ταξινομημένες μετά τον υπολογισμό της εντροπίας

2.10 Σύνοψη και Σύγκριση

Στο κεφάλαιο αυτό αναλύθηκε η έννοια της κορυφογραμμής καθώς και ειπώθηκαν όλα όσα χρειάζονται για την κατανόηση των ερωτήσεων και επερωτήσεων τους ώστε να μπορέσει να λειτουργήσει ένας αλγόριθμος. Επίσης αναλύθηκαν εξειδικευμένοι αλγόριθμοι, οι οποίοι είναι χτισμένοι για να μπορούν να δουλεύουν με ερωτήσεις κορυφογραμμών καθώς και έγινε επιστημονική αφήγηση για τη λειτουργία τους. Κλείνοντας το κεφάλαιο αυτό, θα ακολουθήσει σύγκριση όλων των αλγορίθμων ώστε να υπάρξει μία συνολική εικόνα των δυνατοτήτων τους.

Αλγόριθμος	Πρόοδος	Ψεύτικα σημεία κορυφογραμμής	Χαριστικά σημεία κορυφογραμμής	Προτιμήσεις του χρήστη	Προσαρμογή
D&C	-	✓	✓	-	✓
BNL	-	-	✓	-	✓
BBS	✓	✓	✓	✓	✓
NN	✓	✓	✓	✓	✓
Bitmap	✓	✓	✓	-	✓
SFS	✓	✓	✓	-	✓

Πίνακας 5: Πλεονεκτήματα και μειονεκτήματα αλγορίθμων

Ο παραπάνω πίνακα φαίνεται η κατοχή των σημαντικών χαρακτηριστικών τα οποία περιμένει ο χρήστης από τον αλγόριθμο, τον οποίο και θα χρησιμοποιήσει. Για την περισσότερη κατανόηση αυτών των χαρακτηριστικών:

- Πρόοδος: Το πόσο άμμεσα τα τελικά σημεία κορυφογραμμής θα επιστραφούν αλλά και πρόοδος των υπόλοιπων σημείων, όπου και επιστρέφονται σταδιακά.

- Ψεύτικα σημεία κορυφογραμμής: Με τον όρο αυτό εννοείται ότι υπάρχουν αλγόριθμοι, οι οποίοι επιστρέφουν προσωρινά σημεία κορυφογραμμής καθώς στο τέλος της διαδικασίας τους τα απορρίπτουν.

- Χαριστικά σημεία κορυφογραμμής: Πολλά σημεία οι αλγόριθμοι τα ευνοούν, εφόσον είναι αρκετά καλά σε μία διάσταση τους και δεν επιτυγχάνεται ακριβής σύγκριση.

- Προτιμήσεις του χρήστη: Ο χρήστης να μπορεί να κάνει προτιμήσεις στην σειρά που επιστρέφονται τα σημεία κορυφογραμμής, ενώ ο αλγόριθμος είναι ακόμη σε λειτουργία.

- Προσαρμογή: Το χαρακτηριστικό αυτό φανερώνει το πόσο εύκολα ενσωματώνεται ο αλγόριθμος σε μία υπάρχον βάση δεδομένων.

3 *Βιβλιογραφική Ανασκόπηση*

3.1 Ιστορική αναδρομή και βασικοί αλγόριθμοι

Ο ορισμός των ερωτήσεων κορυφογραμμής σε πολυδιάστατα σύνολα δεδομένων είναι ταυτόσημος με το γνωστό πρόβλημα του μέγιστου διανύσματος (Kung, Luccio, & Preparata, 1975) που υπήρχε από πολύ νωρίς σαν μαθηματικό αλγοριθμικό πρόβλημα. Ο όρος «τελεστής skyline» (skyline operator) εμφανίστηκε για πρώτη φορά σαν πρόταση για την επέκταση των συστημάτων βάσεων δεδομένων (Borzsony, Kossmann, & Stocker, The Skyline operator, 2001). Ο υπολογισμός της κορυφογραμμής γίνεται πολύ πιο αποδοτικά, εν αντιθέσει με εμφωλευμένους βρόγχους επανάληψης, με τη χρήση του BNL (Block-Nested Loops) που αξιοποιεί μία προσωρινή μνήμη (window) (Borzsony, Kossmann, & Stocker, The Skyline operator, 2001). Άλλος ένας χρήσιμος αλγόριθμος είναι ο γνωστός «Διαίρει και βασίλευε» (Divide and Conquer) (Stojmenović & Miyakawa, 1988) που βελτιώνει τον υπολογισμό στις χειρότερες περιπτώσεις αλλά χειροτερεύει στις καλύτερες περιπτώσεις με πολυπλοκότητα $O(n^* (\log n)d-2) + O(n * \log n)$ (Preparata & Shamos, 1985). Επίσης ένας από τους πιο ευρέως γνωστούς αλγόριθμους που βρίσκει εφαρμογή στις ερωτήσεις κορυφογραμμών είναι αυτός του πλησιέστερου γείτονα (NN: Nearest Neighbor) που αξιοποιεί τη δομή δεδομένων R-Tree για να βρει γείτονες πιο κοντά στη βέλτιστη επιλογή και αποκλείει ολόκληρες περιοχές επιλογών (Roussopoulos, Kelley, & Vincent, 1995), (Hjaltason & Samet, 1999). Τέλος, ο BBS (Branch and Bound Skyline) (Papadias, Tao, Fu, & Seeger, 2003) μοιάζει με τον NN αλλά χωρίζει περιοχές αναζήτησης διαφορετικά όπως αναφέραμε στο προηγούμενο κεφάλαιο. Ο BBS τείνει να είναι αρκετά αποδοτικός σε μία βασική παραλλαγή των ερωτήσεων κορυφογραμμής, τις περιορισμένες ερωτήσεις κορυφογραμμής (constrained skyline queries) (Ferhatosmanoglu, Stanoi, Agrawal, & Abbadi, 2001) όπου το σύνολο των επιλογών επιδέχεται περικοπή με βάση κάποιους περιορισμούς που θέτουμε κατά τη δημιουργία του ερωτήματος. Άλλη μία βασική παραλλαγή είναι οι χωρικές ερωτήσεις κορυφογραμμής την οποία θα δούμε αναλυτικότερα παρακάτω (Sharifzadeh & Shahabi, 2006).

3.2 Οι χωρικές επερωτήσεις κορυφογραμμής

Οι Sharifzadeh και Shahabi (2006) αναφέρουν ότι για πρώτη φορά κάποιος παρουσιάζει τις χωρικές επερωτήσεις κορυφογραμμής (SSQ: Spatial Skyline Queries). Παρακάτω παρουσιάζονται οι λύσεις που περιγράφουν στο πρόβλημα των που προσπαθούν να λύσουν.

3.2.1 B2S2: Χωρικός αλγόριθμος Branch & Bound

Παρουσιάζεται μια μέθοδος για την ανίχνευση της περίπτωσης που η έξοδος μιας διαδικασίας είναι επιρρεπής σε αλλαγές. Οι συγγραφείς προτείνουν τη χρήση μιας δομής δεδομένων που ονομάζεται "σύνολο αποκλεισμένων τιμών" (exclusion set) για να καταγράφουν τις τιμές που έχουν αποκλειστεί από τη διαδικασία. Στη συνέχεια, περιγράφεται η λειτουργία του μηχανισμού ανίχνευσης, που ελέγχει αν η τρέχουσα έξοδος της διαδικασίας βρίσκεται σε σύγκρουση με τις προηγούμενες αποκλεισμένες τιμές. Αυτός

ο μηχανισμός μπορεί να ανακαλύψει ανεπιθύμητες αλλαγές και να παράγει ειδοποιήσεις, ενδεχομένως οδηγώντας σε αντίδραση ή επαναφορά της διαδικασίας. Συνολικά, η προτεινόμενη μέθοδος ανίχνευσης επιτρέπει την εξαγωγή αξιόπιστων αποτελεσμάτων από αλληλουχίες επερωτήσεων.

3.2.2 VS2: Αλγόριθμος βασισμένος σε διαγράμματα Voronoi

Μία άλλη μέθοδος που παρουσιάζεται για τη βελτιστοποίηση της ανίχνευσης συγκρούσεων στις εξαγόμενες εντολές ενός συστήματος βάσεων δεδομένων. Οι συγγραφείς προτείνουν τη χρήση ενός συστήματος αυτόματης επανεξέτασης των εξαγόμενων εντολών προκειμένου να μειώσουν τον αριθμό των αναγκαίων ελέγχων συγκρούσεων. Οι εξαγόμενες εντολές του συστήματος βάσεων δεδομένων αξιολογούνται κατάλληλα, επιτρέποντας την αποφυγή περιττών ελέγχων συγκρούσεων. Αυτή η μέθοδος οδηγεί σε αποδοτικότερη ανίχνευση συγκρούσεων και μειώνει τον χρόνο επεξεργασίας των εντολών. Τέλος, παρουσιάζονται πειραματικά αποτελέσματα για την αξιολόγηση της προτεινόμενης μεθόδου και αποδεικνύεται η αποτελεσματικότητα της σε σύγκριση με άλλες μεθόδους ανίχνευσης συγκρούσεων.

Αλγόριθμος VS^2

01. compute the convex hull CH(Q);
02. set S(Q) = {};
03. Heap H = {(NN(q1), mindist(NN(q1),CHv(Q)))};
04. set V isited = {NN(q1)}; set Extracted = {};
05. box B = MBR(SR(NN(q1), Q));
06. while H is not empty 07. (p, key) = first entry of H;
08. if p ∈ Extracted
09. remove (p, key) from H;
10. if p is inside CH(Q) or
11. p is not dominated by S(Q)
12. add p to S(Q);
13. B = B ∩ MBR(SR(p, Q));
14. else
15. add p to Extracted;
16. if S(Q) = ∅ or a Voronoi neighbor of p is in S(Q) 17. for each Voronoi neighbor of p such as p
18. if p ∈ Visited, discard p;
19. if p is inside B or V C(p) intersects with B
20. add p to Visited;
21. add (p, mindist(p,CHv(Q))) to H;
22. return S(Q);

Αλγόριθμος 7: Αλγόριθμος VS2, βασισμένος σε διαγράμματα Voronoi

3.2.3 VCS^2: Αλγόριθμος συνεχών επερωτήσεων βασισμένων σε διαγράμματα Voronoi

Υποθέστε ότι η θέση των σημείων του ερωτήματος αλλάζει. Για την επίλυση αυτού του προβλήματος προτείνεται ο VCS2 ο οποίος βασίζεται σε διαγράμματα Voronoi για συνεχείς SSQ. Ο VCS2 είναι πιο αποδοτικός από την αφελή προσέγγιση καθώς αποφεύγει τους υπερβολικούς και περιττούς ελέγχους υπεροχής. Ο αλγόριθμος διασχίζει το γράφο Delaunay των σημείων όπως ο VS2. Αρχικά, υπολογίζει το κυρτό περίβλημα του πιο πρόσφατου συνόλου αναζήτησης. Στη συνέχεια, το συγκρίνει με το παλιό κυρτό περίβλημα. Ανάλογα με τον τρόπο που διαφέρουν, ο VCS2 αποφασίζει είτε να διασχίσει μόνο συγκεκριμένα τμήματα του γράφου και να ενημερώσει την παλιά κορυφογραμμή, είτε να εκτελέσει ξανά τον VS2 και να δημιουργήσει μια νέα. Στην πρώτη περίπτωση, προσπαθεί να εξετάσει μόνο τα σημεία στο γράφο που αλλάζουν στην υπεροχή λόγω της μετακίνησης. Πρώτα απαριθμούμε τις διάφορες αλλαγές που μπορεί να υποστεί το κυρτό περίβλημα των σημείων όταν ένα σημείο μετακινείται. Γενικά, η αναγνώριση του κατάλληλου μοτίβου μέσω της σύγκρισης των σημείων δεν είναι μια απλή διαδικασία. Επομένως, ο VCS2 προσπαθεί μόνο να αναγνωρίσει συγκεκριμένα απλά μοτίβα και στη συνέχεια να ενημερώσει την κορυφογραμμή ανάλογα. Για κάποια μοτίβα αλλαγής εκτελείται ξανά τον VS2, για άλλα μοτίβα που μπορεί, ο VCS2 διασχίζει συγκεκριμένα τμήματα του γράφου Delaunay. Τέλος, ο VS2 διασχίζει το υπόλοιπο τμήμα του γράφου Delaunay για να ενημερώσει την κορυφογραμμή στις υπόλοιπες περιοχές που δεν καλύπτονται από τα παραπάνω μοτίβα.

3.2.4 Συμπεράσματα των Sharifzadeh και Shahabi

Σε αυτή τους τη δημοσίευση οι Sharifzadeh και Shahabi αποδεικνύουν αναλυτικά ότι υπάρχει ένα σύνολο καθορισμένων σημείων κορυφογραμμής, των οποίων τα κελιά Voronoi βρίσκονται μέσα ή τέμνονται με το κυρτό κύτος των σημείων του ερωτήματος. Αποδείξαμε επίσης ότι οι θέσεις αυτών των σημείων του ερωτήματος μέσα στο κυρτό κύτος όλων των σημείων του ερωτήματος δεν έχουν καμία επίδραση στην τελική χωρική κορυφογραμμή. Με βάση αυτά τα θεωρητικά ευρήματα, πρότειναν δύο αποδοτικούς αλγόριθμους για SSQ λαμβάνοντας υπόψη τα στατικά σημεία του ερωτήματος. Μέσα από εκτεταμένα πειράματα, έδειξαν την υπεροχή των αλγορίθμων τους έναντι της παραδοσιακής προσέγγισης του BBS. Ο αλγόριθμός VS^2 παρουσιάζει έως και 6 φορές ταχύτερη βελτίωση απόδοσης σε σχέση με το BBS. Μελέτησαν επίσης μια παραλλαγή του προβλήματος των SSQ με μετακίνηση των σημείων του ερωτήματος. Ο αλγόριθμός VCS^2 ενημερώνει αποτελεσματικά την κορυφογραμμή με οποιαδήποτε αλλαγή στις θέσεις των σημείων του ερωτήματος. Η προσέγγισή αυτή παρουσιάζει σημαντικά καλύτερη απόδοση από τον εκ νέου υπολογισμό της κορυφογραμμής, ακόμη και χρησιμοποιώντας τον αποτελεσματικό αλγόριθμό VS^2. Τέλος, έδειξαν πώς και οι τρεις προτεινόμενοι αλγόριθμοι μπορούν να αντιμετωπίσουν το πρόβλημα των SSQ όταν αναμιγνύονται με μη χωρικά χαρακτηριστικά. Ως σκοπό τους για επόμενες έρευνες έθεσαν τη μελέτη των προκλήσεων των SSQ σε μετρικούς χώρους όπως τα οδικά δίκτυα καθώς θα χρησιμοποιήσουν αυτές τις ιδιότητες για να προτείνουν αποτελεσματικούς αλγόριθμους SSQ για βάσεις δεδομένων οδικών δικτύων.

3.3 Πολυεπίπεδες επερωτήσεις κορυφογραμμών

Οι Kodama, Iijima, Guo και Ishikawa στη δημοσίευσή τους Skyline Queries Based on User Locations and Preferences for Making Location-Based Recommendations (2009) παρουσιάζουν μια προσέγγιση για τη πρόταση σημείων ενδιαφέροντος, όπως εστιατόρια, σε έναν χρήστη κινητής συσκευής, λαμβάνοντας υπόψη την τρέχουσα τοποθεσία και τις προτιμήσεις του. Στο πλαίσιό αυτό, ο χρήστης παρέχει αρχικά ένα προφίλ,

το οποίο καταγράφει τις προτιμήσεις του, όπως τύποι φαγητού και τιμές. Στη συνέχεια, επιλέγονται τα σημεία που θα προταθούν από τη βάση δεδομένων βασιζόμενοι στο προφίλ του χρήστη καθώς και την τρέχουσα τοποθεσία του. Για να εξαχθούν καλά αποτελέσματα, επεκτείνεται η έννοια των επερωτήσεων κορυφογραμμής χωρικών στοιχείων συμπεριλαμβάνοντας όχι μόνο πληροφορίες απόστασης αλλά και κατηγορικές πληροφορίες προτίμησης.

Αλγόριθμος 1: Κορυφογραμμής χωρικών δεδομένων με προτιμήσεις

```
1: function SSP(p, q)
2:                                  // p: user's profile, q: user's location
3:      S ← ∅                       // Skyline objects
4:      T ← get category table(p)
5:      NN_init(q)                  // Initialize nearest neighbor query
6:      repeat
7:          o ← NN fetch( )         // Get next nearest object
8:          if match(T,o) then
9:              // o is a skyline object
10:             S ← S ∪ {o}
11:             Remove the matched entry e ∈ T from T
12:             Remove all the entries e ∈ T such that o ≺ e
13:         end if
14:     until T is empty
15:     return S
16: end function
```

Αλγόριθμος 8: Ψευδοκώδικας κορυφογραμμής χωρικών δεδομένων με προτιμήσεις

Παρουσιάζεται ο αλγορίθμος 1, που είναι ένας αλγόριθμος αναζήτησης κορυφογραμμής που παρέχει καλά στοιχεία βάσει της σχέσης κυριαρχίας. Ο αλγόριθμος λειτουργεί καλά, αλλά έχει το πρόβλημα ότι η παραγόμενη κορυφογραμμή μπορεί να αποτελείται από ένα μικρό αριθμό σημείων. Αυτό συμβαίνει όταν ένα ή δύο σημεία είναι πολύ ισχυρά σε σύγκριση με άλλα αντικείμενα ή όταν ο αριθμός των κατηγοριών είναι μικρός. Ένας χρήστης που θέλει να συγκρίνει αρκετούς υποψήφιους δεν θα ικανοποιηθεί από ένα τέτοιο αποτέλεσμα. Ως λύση για αυτό το πρόβλημα, προτείνεται η ιδέα των πολυεπίπεδων αναζητήσεων κορυφογραμμής. Αποκαλούμε τα αντικείμενα κορυφογραμμής που έχουν ληφθεί από τον αλγόριθμο στην προηγούμενη ενότητα τα αντικείμενα κορυφογραμμής πρώτου επιπέδου. Εάν ο αριθμός των σημείων πρώτου επιπέδου είναι μικρότερος από το k, που είναι ένας αριθμός που καθορίζεται από τον χρήστη, υπολογίζουμε τα σημεία δεύτερου επιπέδου. Για αυτόν τον υπολογισμό, αποκλείονται τα αντικείμενα πρώτου επιπέδου από την εξέταση. Για παράδειγμα, αν υποθέσουμε ότι ο χρήστης χρειάζεται τουλάχιστον πέντε αντικείμενα και ορίζει k = 5. Χρησιμοποιώντας τον αλγόριθμο αναζήτησης κορυφογραμμής που παρουσιάζεται παραπάνω, λαμβάνουμε το πρώτο επίπεδο κορυφογραμμής S1 = {b, d, g}. Εφόσον ο αριθμός είναι μικρότερος από το κατώφλι k, γίνεται προσπάθεια να βρεθούν τα αντικείμενα δεύτερου επιπέδου αποκλείοντας τα αντικείμενα του S1 από υποψήφια. Χρησιμοποιώντας τον ίδιο αλγόριθμο, λαμβάνονται τα σημεία δεύτερου επιπέδου: σε αυτήν την περίπτωση, S2 = {e, h, j}. Εφόσον λαμβάνουμε έξι σημεία

συνολικά, μπορούμε να τερματίσουμε τη διαδικασία εδώ. Εάν ο συνολικός αριθμός ήταν ακόμα μικρότερος από το k, θα συνεχίζαμε με την επιλογή στην τρίτη επίπεδο. Η Εικόνα 6 δείχνει τα προκύπτοντα σημεία. Ένα αστέρι και ένας κύκλος αντιπροσωπεύουν τα σημεία πρώτου και δεύτερου επιπέδου, αντίστοιχα. Ο αλγόριθμος 2 δείχνει τον αλγόριθμο πολυεπίπεδης χωρικής αναζήτησης κορυφογραμμής. Ο αλγόριθμος είναι απλός και το αποτέλεσμα S περιέχει τα σύνολα σημεία για διάφορα επίπεδα (S = {S1, S2,...}).

4 *Υλοποίηση και Γραφική διεπαφή*

4.1 Υλοποίηση του Block Nested Loop με την χρήση Java

Μέσα από την γνώση και την μελέτη των αλγορίθμων οι οποίοι συνέβαλαν στην υλοποίηση καινούργιων και πιο εξιδεικευμένων αλγορίθμων πάνω στην μελέτη παραλλαγών ερωτήσεων κορυφογραμμών του προηγούμενου κεφαλαίου, έρχεται η στιγμή για την υλοποίηση ενός αλγορίθμου μέσα από τον οποίο ο χρήστης θα είναι σε θέση να μπορεί να συλλέξει αποτελέσματα κορυφογραμμών άμεσα και γρήγορα.

Το κεφάλαιο αυτό επικεντρώνεται στις κλάσεις που χρησιμοποιήθηκαν για την υλοποίηση του αλγόριθμου Block Nested Loop (BNL) στην γλώσσα προγραμματισμού Java. Το πρόγραμμά είναι ικανό να δέχεται αντικείμενα ως εισροή (συγκεκριμένα στην παρούσα διπλωματική γίνεται προσπέλαση του προγράμματος με αποθηκευμένες εγγραφές, οι οποίες παρέχουν αυτοκίνητα και έχουν πολλούς παραμέτρους το καθένα) και μέσα από πολλά κριτήρια όπου μπορεί να θέτει ο χρήστης, του επιστρέφεται η κορυφογραμμή μέσα από την οποία θα μπορεί να αντλεί τα πιο αποδοτικά και βέλτιστα αποτελέσματα.

Ο αλγόριθμος ολοκληρώνεται με εννέα κλάσεις οι οποίες αναλύονται διεξοδικά βήμα, βήμα καθώς και υπάρχει αναλυτική περιγραφή για το γραφικό περιβάλλον του αλγορίθμου προσφέροντας στον χρήστη μία πιο εύχρηστη και γρήγορη χρήση, αλλά και οπτικά αποτελέσματα τα οποία είναι περισσότερο της αρέσκειας του.

4.1.1 Κλάση Field

Η κλάση Field αντιπροσωπεύει τα πεδία που έχουν ληφθεί από το csv που διαβάζουμε κάθε φορά. Οι ιδιότητες που περιέχει είναι:

1. name: Το όνομα του πεδίου.
2. displayed: Boolean για το αν θέλουμε να εμφανίσουμε το πεδίο στο τελικό αποτέλεσμα.
3. calculatable: Boolean για το αν θέλουμε το πεδίο να χρησιμοποιηθεί στους υπολογισμούς κυριαρχίας.
4. ascending: Boolean για το αν θέλουμε στους υπολογισμούς κυριαρχίας να είναι οι μεγάλες ή οι μικρές τιμές κυρίαρχες.

Η κλάση Field περιλαμβάνει δύο constructors:

1. Ένα constructor που χρησιμεύει στην αρχικοποίηση του πεδίου skyline_id με συγκεκριμένες τιμές.
2. Ένα constructor που χρησιμεύει στην αρχικοποίηση των υπόλοιπων πεδίων με βάση τα δεδομένα που λήφθηκαν από το csv και από το χρήστη.

4.1.2 Κλάση Entry

Η κλάση Entry αντιπροσωπεύει τις ξεχωριστές εγγραφές στις οποίες διατηρούμε τιμές των αντίστοιχων πεδίων. Έχουμε επιλέξει να διαχειριζόμαστε τις τιμές σε ένα LinkedHashMap γιατί μας ενδιαφέρει η σειρά με την οποία διατηρούνται τα στοιχεία και θέλουμε να έχουμε πρόσβαση σε αυτά με το όνομα του πεδίο ενώ ταυτόχρονα είναι μία από τις πιο γρήγορες δομές δεδομένων που παρέχει η Java. Η επιλογή να διατηρούμε τις τιμές σε μία δομή δεδομένων αντί να ορίσουμε τις τιμές μέσα στην κλάση ως ιδιότητες έρχεται από την ανάγκη να μπορούμε να διαχειριστούμε διαφορετικά datasets με διαφορετικά πεδία κάθε φορά. Ο constructor είναι φτιαγμένος να αρχικοποιεί την ιδιότητα values αλλά χρησιμοποιούμε δύο υπερφορτωμένες (overloaded) στατικές μεθόδους, ονόματι isolateColumns, για την δημιουργία αντικειμένων της Entry. Απαραίτητες παράμετροι είναι:

1. Ένα αντικείμενο της κλάσης CSVRecord της βιβλιοθήκης Apache Commons CSV, η οποία χρησιμοποιείται ευρέως στο έργο για την ανάγνωση και την εγγραφή των csv αρχείων, το οποίο περιέχει τις τιμές μίας γραμμής που αναγνώστηκε από κάποιο αρχείο csv.
5. Ένα ArrayList από αντικείμενα της κλάσης Field, που περιέχει τα πεδία τα οποία έχει επιλέξει ο χρήστης να χρησιμοποιήσει στους υπολογισμούς του.

Η δεύτερη isolateColumns δέχεται και έναν ακέραιο newId με σκοπό να τον καταχωρήσει στο πεδίο skyline_id. Στην ουσία η δεύτερη isolateColumns χρησιμοποιείται με records από το αρχικό csv ενώ η πρώτη από το parsed csv όπου εκεί υπάρχει ήδη το πεδίο skyline_id αφού περιέχει εγγραφές που προήλθαν από την δεύτερη isolateColumns. Η μέθοδος hasNKeys σκοπό έχει τη σύγκριση του αριθμού των πεδίων της εγγραφής και των πεδίων που έχει επιλέξει ο χρήστης. Αν δεν συμφωνούν σημαίνει ότι κάποιο ή κάποια πεδία λείπουν από την εγγραφή άρα δεν την αξιοποιούμε. Η συνάρτηση get έχει σκοπό να μας διευκολύνει να λάβουμε την τιμή που επιθυμούμε από την ιδιότητα values. Εν συνεχεία, η συνάρτηση isDominatedBy δέχεται μία άλλη εγγραφή και τα πεδία που χρησιμοποιούνται στον υπολογισμό της κυριαρχίας για να τα συγκρίνει. Θα πρέπει να μας επιστρέφει αληθές μόνο αν σε όλα τα πεδία η άλλη εγγραφή είναι κυρίαρχη ή σε όλα τα πεδία έχουν ίδιες τιμές αλλά σε τουλάχιστον ένα είναι κυρίαρχη η άλλη. Για αυτό τον υπολογισμό χρησιμοποιείται και η μέθοδος isDominatedInFieldBy που σκοπό έχει την επιμέρους σύγκριση τιμή για ένα πεδίο κάθε φορά. Δύο αξιοσημείωτοι μηχανισμοί είναι:

1. totalDomination και fieldDomination: Με αυτές τις μεταβλητές και τον ακέραιο που επιστρέφεται από την isDominatedInFieldBy έχουμε δημιουργήσει ένα μηχανισμό όπου όταν σε ένα πεδίο έχουμε ισότητα επιστρέφεται 0, όταν είναι κυρίαρχη η άλλη τιμή επιστρέφεται 1, ενώ αν κυρίαρχη είναι η τιμή της τρέχουσας εγγραφής επιστρέφεται -1. Στην περίπτωση του -1 έχουμε σίγουρο False στο αν είναι κυρίαρχη η άλλη εγγραφή. Στην περίπτωση του 0 και 1 τα προσθέτουμε στην totalDomination έτσι ώστε όταν ολοκληρωθεί ο υπολογισμός να ξέρουμε ότι σε τουλάχιστον ένα πεδίο η άλλη εγγραφή είναι κυρίαρχη της τωρινής.
2. Field.ascending: Όπως είδαμε, η κλάση Field περιέχει μία ιδιότητα με όνομα ascending η οποία μας δείχνει αν στο συγκεκριμένο πεδίο είναι κυρίαρχες οι μεγάλες τιμές ή οι μικρές και το οποίο χρησιμοποιούμε στους υπολογισμούς κατάλληλα.

4.1.3 Κλάση Skyline

Η κλάση Skyline και κατ' επέκταση η μέθοδός της main αποτελούν την αρχή της εφαρμογής. Η λειτουργία τους είναι να καλούν τα πιο βασικά modules του προγράμματος τα οποία με τη σειρά τους θα αξιοποιήσουν όλες τις δυνατότητες που προσφέρονται από την εφαρμογή. Πιο αναλυτικά:

1. Εκτελείται η επιλογή αρχείου csv
2. Αρχίζει η εκτέλεση της λειτουργικότητας του CSVParser που θα δούμε παρακάτω.
3. Αρχίζει η εκτέλεση του BNL.

4.1.4 Κλάση Configuration

Η κλάση Configuration έχει διπλό χαρακτήρα, χρησιμεύει στην διαχείριση των επιλογών του χρήστη για τα πεδία εσωτερικά στο πρόγραμμα αλλά αποτελεί και διεπαφή χρήστη μέσω γραφικού περιβάλλοντος για να λάβει τις επιλογές αυτές. Διατηρεί μία ιδιότητα με όνομα fields τύπου ArrayList από αντικείμενα της κλάσης Field. Ο constructor της λαμβάνει τα ονόματα των πεδίων που έχουμε λάβει από την πρώτη γραμμή του εισαγόμενου αρχείου csv και δημιουργεί ένα γραφικό περιβάλλον όπου για το κάθε ένα από αυτά ο χρήστης έχει τη δυνατότητα να επιλέξει τις ιδιότητές τους όπως αυτές περιγράφονται στην κλάση Field. Όταν ο χρήστης ρυθμίσει κατάλληλα την εφαρμογή ανάλογα με το ποια πεδία θέλει μόνο να εμφανιστούν στο τελικό αρχείο, ποια θέλει να χρησιμοποιηθούν στους υπολογισμούς και με ποιο τρόπο (αύξουσα/φθίνουσα), κάνει κλικ στο κουμπί Calculate και τότε προστίθεται στο προαναφερόμενο fields το πεδίο skyline_id που έχουμε αναφέρει προηγουμένως και όλα τα πεδία με τις επιλογές του χρήστη ως ιδιότητες. Έπειτα αυτή η κλάση έχει τη δυνατότητα να προσφέρει μέσω μεθόδων:

1. getFields: Το σύνολο των πεδίων σε ένα ArrayList από αντικείμενα της κλάσης Field.
2. getUsefulFields: Ένα ArrayList από αντικείμενα της κλάσης Field μόνο με τα «χρήσιμα» πεδία, δηλαδή όσα πρόκειται να εμφανιστούν στο τελικό αρχείο ή έχουν επιλεχθεί να συμμετέχουν στους υπολογισμούς κυριαρχίας.
3. getFieldsNames: Ένα ArrayList από αντικείμενα της κλάσης String με τα ονόματα όλων των πεδίων.
4. getUsefulFieldsNames: Ένα ArrayList από αντικείμενα της κλάσης String με τα ονόματα όλων των «χρήσιμων» πεδίων.
5. getCalculatableFields: Ένα ArrayList από αντικείμενα της κλάσης Field μόνο με τα πεδία που έχουν επιλεχθεί να συμμετέχουν στους υπολογισμούς κυριαρχίας.
6. noFields: Boolean για το αν δεν υπάρχουν περασμένα πεδία

4.1.5 Κλάση CSVParser

Η κλάση CSVParser είναι υπεύθυνη για την αρχική προσπέλαση του αρχείου CSV που θα εισάγει ο χρήστης ως είσοδο προς επεξεργασία. Στον constructor δίνουμε το μονοπάτι του αρχείου που επίλεξε ο χρήστης και αυτός το αποθηκεύει στο αντικείμενο και δημιουργεί το μονοπάτι του αρχείου που θα εξάγει (parsed). Έπειτα η συνάρτηση startProcess αναλαμβάνει να διαβάσει την πρώτη γραμμή με χρήση της

βιβλιοθήκης Apache Commonos CSV η οποία περιέχει τα πεδία των δεδομένων και να δημιουργήσει ένα αντικείμενο της κλάσης Configuration περνώντας του μία λίστα με τα ονόματα των πεδίων. Η configuration όπως είπαμε προηγουμένως θα εμφανίσει ένα γραφικό περιβάλλον στο χρήστη με επιλογές για το ποια πεδία θα χρησιμοποιηθούν στους υπολογισμούς και με ποιο τρόπο. Μόλις ολοκληρωθεί αυτή η διαδικασία, ξεκινάει η προσπέλαση του αρχείου όπου για κάθε γραμμή (record) δημιουργούμε ένα αντικείμενο της κλάσης Entry με χρήση της στατικής μεθόδου isolateColumns και ελέγχουμε αν η εγγραφή έχει όσες τιμές όσα και τα «χρήσιμα» πεδία έχει επιλέξει ο χρήστης. Αν δεν έχει όσες τιμές απαιτούνται, δεν την εισάγουμε στο εξαγώγιμο αρχείο, άρα και δεν την λαμβάνουμε υπόψη μας στους υπολογισμούς. Η κλάση προσφέρει άλλες δύο μεθόδους:

1. getOutputFilePath: Επιστρέφει αντικείμενο της κλάσης String με το μονοπάτι του αρχείου που θα εξάγει.
2. getConfig: Επιστρέφει το αντικείμενο της κλάσης Configuration που δημιούργησε.

4.1.6 Κλάση CsvCacher

Η κλάση CsvCacher είναι υπεύθυνη να τροφοδοτεί τον BNL αλγόριθμο με εγγραφές από την cache που έχει δημιουργηθεί. Στον constructor λαμβάνει το αρχείο από το οποίο θα διαβάζει εγγραφές και το αποθηκεύει στην αντίστοιχη ιδιότητά της. Έπειτα στη μέθοδο startProcess αρχικοποιείται το αντικείμενο της κλάσης CSVFormat της βιβλιοθήκης Apache Commons CSV που χρησιμοποιείται για την ανάγνωση από το αρχείο CSV και ο αντίστοιχος iterator. Με αυτό τον το αντικείμενο της κλάσης CsvCacher που δημιουργεί ο BNL είναι έτοιμο να παρέχει εγγραφές με κλήση στη μέθοδο nextEntry η οποία χρησιμοποιεί τον iterator για να διαβάσει records αν υπάρχουν από το αρχείο και να επιστρέψει το πρώτο κατάλληλο. Τέλος η μέθοδος endProcess αναλαμβάνει το ασφαλές κλείσιμο του αρχείου.

4.1.7 Κλάση BNL

Η κλάση BNL περιλαμβάνει όλη τη λειτουργικότητα για τον υπολογισμό των βέλτιστων αποτελεσμάτων. Καταρχάς δημιουργεί νέα αρχεία CSV με ονόματα newCache (αρχείο μνήμης cache), tempPath (προσωρινό αρχείο) και result (αρχείο αποτελεσμάτων). Τα αρχεία αυτά αποθηκεύονται στον ίδιο φάκελο με το αρχείο εισαγωγής και το αρχείο parsed. Στην κλάση υπάρχουν κάποιες βοηθητικές συναρτήσεις:

exportResults: Εξάγει το ArrayList resultSkyline που του περνάμε στο αρχείο των αποτελεσμάτων

isDominatedBy: Ελέγχει αν το πρώτο όρισμα είναι dominated από το δεύτερο με βάση τα πεδία που έχει επιλέξει ο χρήστης.

newOutputFrom: Χρησιμοποιείται για να ανοίξει μία καινούρια ροή εξαγωγής στο αρχείο που περνιέται σαν όρισμα.

newInputFrom: Χρησιμοποιείται για να ανοίξει μία καινούρια ροή εισαγωγής από το αρχείο που περνιέται σαν όρισμα.

Ο αλγόριθμος στην αρχή της εκτέλεσής του διαβάζει από το αρχείο parsed μία εγγραφή και την περνάει κατευθείαν στο window που αποτελεί το παράθυρο του BNL και στο οποίο αποθηκεύονται εγγραφές που δεν μπορούν να συγκριθούν μεταξύ τους ξεχωριστά και στο τέλος της κάθε επανάληψης αποτελούν σημεία της κορυφογραμμής. Στη συνέχεια ο βρόγχος της επανάληψης ξεκινάει ξανά από την αρχή.

Ο αλγόριθμος τώρα που δεν έχει κενό παράθυρο, διαβάζει εγγραφές από το αρχείο parsed και ελέγχει αν είναι η εγγραφή που μόλις διαβάσαμε κυριαρχείται από κάποια του παραθύρου. Αν κυριαρχείται από κάποια του παραθύρου, διαβάζουμε την επόμενη εγγραφή και εκτελούμε αντίστοιχους ελέγχους. Αν όμως η νέα εγγραφή κυριαρχεί κάποια του παραθύρου, τότε η εγγραφή του παράθυρου αφαιρείται από το παράθυρο. Αν στο τέλος του ελέγχου δεν κυριαρχείται από καμία εγγραφή του παραθύρου, τότε περνάμε την νέα εγγραφή που διαβάσαμε στο παράθυρο αν δεν έχει φτάσει το μέγιστο αριθμό εγγραφών που του έχουμε ορίσει, αλλιώς την περνάμε στο προσωρινό αρχείο και ορίζουμε να περνάνε όλες οι εγγραφές στο προσωρινό αρχείο μέχρι να αδειάσουμε το παράθυρο.

Όταν ολοκληρωθεί η ανάγνωση από το αρχείο parsed, περνάμε όλες τις εγγραφές του παραθύρου στο αποτέλεσμα. Αν έχουμε εγγραφές στο προσωρινό αρχείο, αρχικοποιούμε νέο παράθυρο και ξεκινάμε να εκτελούμε την ίδια διαδικασία αλλά αυτή τη φορά διαβάζουμε εγγραφές από το προσωρινό αρχείο αντί του parsed. Μόλις ολοκληρωθεί και αυτή η διαδικασία το αποτέλεσμά μας είναι έτοιμο και προχωράμε στην εγγραφή του αρχείου result και στην εμφάνιση των στατιστικών στοιχείων σχετικά με τους χρόνους εκτέλεσης και τον χώρο αποθήκευσης που καταλήφθηκε για τους υπολογισμούς.

4.1.8 Κλάση SFile

Η κλάση SFile περιλαμβάνει μεθόδους που εκτελούν εργασίες σχετικές με την ανάγνωση και την εγγραφή αρχείων CSV.

- getReader(): Επιστρέφει το αντικείμενο της κλάσης BufferedReader με το οποίο μπορούμε να διαβάζουμε από μία ροή. Αυτό το χρησιμοποιεί η CSVFormat για να ξέρει από που να διαβάσει.
- getWriter(): Επιστρέφει το αντικείμενο της κλάσης BufferedWriter με το οποίο μπορούμε να γράψουμε σε μία ροή. Αυτό το χρησιμοποιεί η CSVPrinter για να ξέρει που να γράψει.
- openRFile(String sFile): Εκκινεί μία ροή εισαγωγής με το αρχείο στο μονοπάτι που του δίνουμε και αρχικοποιώντας ένα αντικείμενο της κλάσης BufferedReader.
- openWFile(String sFile): Εκκινεί μία ροή εξαγωγής με το αρχείο στο μονοπάτι που του δίνουμε και αρχικοποιώντας ένα αντικείμενο της κλάσης BufferedWriter.
- closeRFile(): Κλείνει τη ροή του αρχείου ανάγνωσης.
- closeWFile(): Κλείνει τη ροή του αρχείου εγγραφής.

4.1.9 Κλάση TimeCounter

Η κλάση TimeCounter αποτελεί ένα Mixin, δηλαδή μία κλάση που περιλαμβάνει λειτουργίες που άλλες κλάσεις που την επεκτείνουν, τις κληρονομούν. Η χρησιμότητά της περιλαμβάνει χρονικές μετρήσεις για διάφορα τμήματα κώδικα. Στον constructor της κλάσης αρχικοποιούμε δύο ιδιότητες, αντικείμενα της κλάσης Map που μας δίνουν τη δυνατότητα να αποθηκεύουμε αντικείμενα της κλάσης Long με ένα κλειδί τύπου String για την ανάκτησή τους. Στην ιδιότητα functionStartTime αποθηκεύονται οι χρόνοι έναρξης της κάθε εκτέλεσης και στην functionTimeSum οι συνολικοί χρόνοι εκτέλεσης. Η λογική της κλάσης είναι να αποθηκεύουμε ακριβείς χρόνους που τρέχουν συγκεκριμένα τμήματα κώδικα και να προσαυξάνουμε το συνολικό χρόνο εκτέλεσής τους. Για αυτό και σε όλες τις μεθόδους απαιτείται να δώσουμε ένα αντικείμενο της κλάσης String το οποίο αποτελεί όνομα για το συγκεκριμένο τμήμα κώδικα. Η μέθοδος start ζητάει ένα τέτοιο αντικείμενο με όνομα callerFunctionName και αποθηκεύει το τρέχον χρόνο στην ιδιότητα functionStartTime με κλειδί το callerFunctionName. Η χρήση της start είναι να καλείται όταν ξεκινάει η εκτέλεση του κομματιού κώδικα που θέλουμε να χρονομετρήσουμε. Με την ίδια λογική όταν ολοκληρώνεται η εκτέλεση ενός κομματιού κώδικα που θέλουμε να χρονομετρήσουμε καλούμε την stop. Η μέθοδος stop με τη σειρά της ζητάει και αυτή ένα αντικείμενο της κλάσης String το οποίο θα πρέπει να είναι το ίδιο με αυτό που δώσαμε στην αντίστοιχη start του συγκεκριμένου τμήματος κώδικα. Η stop θα λάβει τον χρόνο έναρξης εκτέλεσης από την functionStartTime, θα αφαιρέσει αυτή την τιμή από το τρέχον χρόνο και θα αποθηκεύσει στην functionTimeSum τη διαφορά στην ιδιότητα functionTimeSum με το ίδιο όνομα κλειδιού. Έτσι έχουμε τον χρόνο εκτέλεσης του συγκεκριμένου τμήματος κώδικα. Έπειτα, αν το ίδιο τμήμα κώδικα εκτελεστεί και πάλι, η start θα αποθηκεύσει νέο χρόνο και η stop αφού υπολογίσει την διαφορά, θα προσαυξήσει την τιμή που υπάρχει στην functionTimeSum και με αυτό τον τρόπο κρατάμε το άθροισμα όλων των χρόνων εκτέλεσης όσες φορές έχει τρέξει το συγκεκριμένο κομμάτι κώδικα. Επίσης διατηρούμε τους χρόνους έναρξης και συνολικού χρόνου εκτέλεσης για πολλαπλά τμήματα κώδικα με τη χρήση των ονομάτων κλειδιών. Τέλος όταν έρθει η ώρα το αντικείμενό μας να πεθάνει, μπορούμε να τρέξουμε τη μέθοδο printResults που θα τυπώσει τους συνολικούς χρόνους εκτέλεσης και το όνομα του κάθε κομματιού κώδικα.

4.2 Αναβάθμιση του αλγορίθμου

Για την αναβάθμιση του αλγόριθμου που αναφέρεται στο προηγούμενο κεφάλαιο με υποστήριξη υπολογισμού απόστασης προστέθηκαν τρία χαρακτηριστικά στην υλοποίησή μας τα οποία περιγράφονται αναλυτικά παρακάτω ανάλογα με τη λειτουργική μονάδα (module) στην οποία υλοποιήθηκαν:

4.2.1 Γραφικό Περιβάλλον

Προστέθηκαν δύο στήλες τύπου μονής επιλογής (radio buttons) κατά την επιλογή των χαρακτηριστικών του ερωτήματος μέσω των οποίων μπορούμε να επιλέξουμε ποια δύο πεδία/στήλες από το εισαγόμενο αρχείο αφορούν το γεωγραφικό μήκος (longitude) και το γεωγραφικό πλάτος (latitude). Όταν πατηθεί το κουμπί Calculate, για να εκκινήσει η επεξεργασία του εισαγόμενου αρχείου και έπειτα ο υπολογισμός της απάντησης του ερωτήματος, το πρόγραμμα θα καταλάβει ότι χρειαζόμαστε υπολογισμό με βάση την απόσταση αφού έχουμε επιλέξει τις στήλες με τις γεωγραφικές συντεταγμένες και θα μας ρωτήσει για την δική μας τοποθεσία με χρήση δύο απλών παραθύρων διαλόγου. Το πρώτο παράθυρο διαλόγου θα μας ρωτήσει για το γεωγραφικό μας μήκος και το δεύτερο για το γεωγραφικό μας πλάτος και θα πρέπει να εισάγουμε τις αντίστοιχες τιμές.

4.2.2 Προεπεξεργασία

Κατά την εκτέλεση της προεπεξεργασίας του εισαγόμενου αρχείου που ξεκινά από την κλάση CSVParser, χρησιμοποιείται η κλάση Entry για την προεπεξεργασία και φιλτράρισμα της κάθε ξεχωριστής εγγραφής όπως περιγράφηκε αναλυτικότερα προηγουμένως. Η κλάση Entry με τη σειρά της εκτελεί τον κατάλληλο διαχωρισμό στηλών με βάση το κάθε επιλεγμένο πεδίο. Αν συναντήσει τα στοιχεία γεωγραφικού μήκους και πλάτους, τότε δημιουργεί από αυτά ένα νέο πεδίο με όνομα _s_distance. Το νέο πεδίο με όνομα _s_distance αντιπροσωπεύει την απόσταση σε μέτρα μεταξύ των γεωγραφικών συντεταγμένων της κάθε εγγραφής με τις γεωγραφικές συντεταγμένες που δώσαμε ως δικές μας στα παράθυρα διαλόγου που περιγράψαμε στην ακριβώς προηγούμενη ενότητα. Για τον υπολογισμό της διαφοράς χρησιμοποιήθηκε η συνάρτηση haversinMeters της κλάσης SloppyMath της βιβλιοθήκης Lucene της Apache η οποία χρησιμοποιεί τον ημιπαρημίτονο (haversine) τύπο υπολογισμού απόστασης. Με αυτό τον τρόπο λαμβάνουμε πολύ γρήγορους και φθηνούς υπολογισμούς που είναι ιδιαίτερα χρήσιμοι στην προεπεξεργασία του εισαγόμενου αρχείου καθώς εκτελείται σχεδόν ανά εγγραφή. Πετυχαίνουμε αυτή την ταχύτητα καθώς με αυτό τον τύπο λαμβάνουμε υπόψη μας την Γη σαν μία σφαίρα και όχι σαν πεπλατυσμένο ελλειψοειδές και αυτό μας αφαιρεί πολλές περίπλοκες πράξεις από τον υπολογισμό με αντάλλαγμα μείωση της ακρίβειας μέχρι και 0.5%. Στη συνέχεια αφού προστέθηκε το πεδίο _s_distance, θα ληφθεί υπόψη και στην εκτέλεση του BNL.

4.2.3 Χάρτης

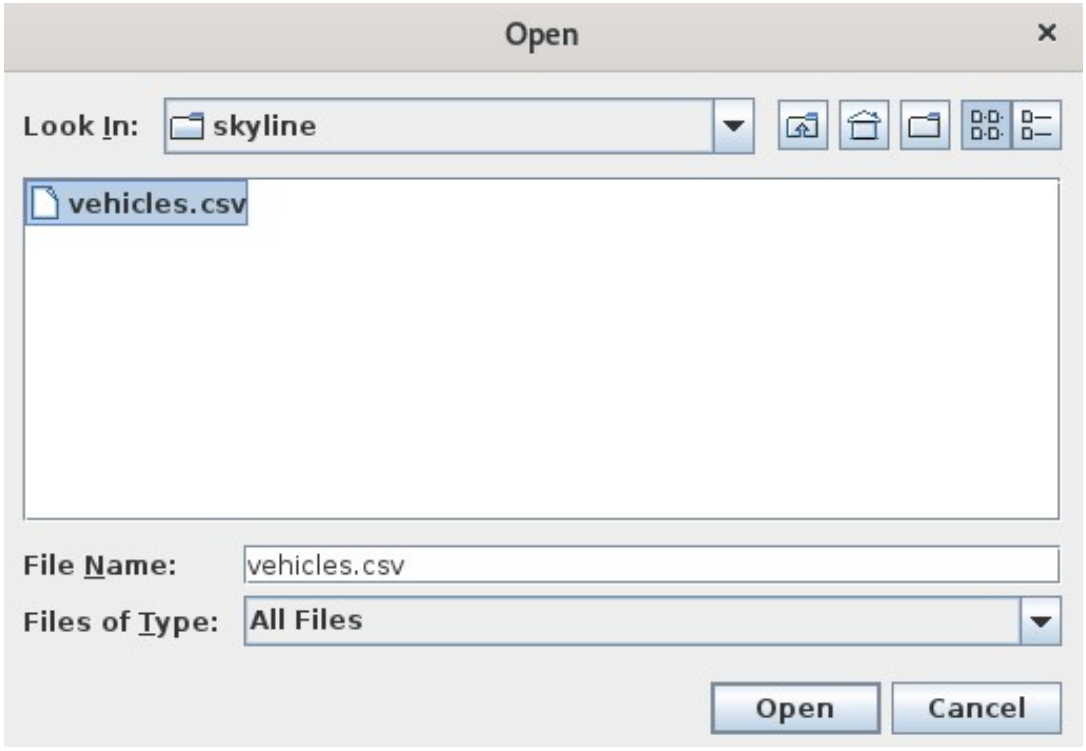
Η τελευταία προσθήκη αφορά την προβολή των αποτελεσμάτων του υπολογισμού με βάση την απόσταση. Χρησιμοποιήθηκε η ανοιχτού κώδικα βιβλιοθήκη JMapViewer του οργανισμού OpenStreetMap με σκοπό την εμφάνιση των αποτελεσμάτων σε γραφικό περιβάλλον διεπαφής χρήστη. Υλοποιήθηκε γραφικό περιβάλλον προβολής χάρτη με δυνατότητες κύλισης, μεγέθυνσης, σμίκρυνσης, προβολή σημείου χρήστη με βάση τις συντεταγμένες που έχουμε εισάγει και προβολή όλων των αποτελεσμάτων με βάση τις συντεταγμένες τους πάνω στο χάρτη. Το γραφικό περιβάλλον του χάρτη θα αναλυθεί παρακάτω στην αντίστοιχη ενότητα.

4.3 Χρήση Γραφικής Διεπαφής Χρήστη (GUI)

Η βιβλιοθήκη javax.swing στη Java χρησιμοποιείται για τη δημιουργία γραφικών διεπαφών χρήστη (GUI: Graphical User Interface) για εφαρμογές με γραφικό περιβάλλον. Χρησιμοποιούνται κυρίως για εφαρμογές desktop, αλλά μπορούν να χρησιμοποιηθούν και σε εφαρμογές web. Η χρησιμότητα της javax.swing στη Java είναι πολλαπλή, καθώς παρέχουν μια ευκολότερη και πιο φιλική διεπαφή για τον χρήστη, καθιστώντας την αλληλεπίδραση με την εφαρμογή πιο απλή και πιο ευχάριστη.

Επίσης, μέσω του GUI, μπορεί να δημιουργηθεί μία εφαρμογή με πολλά διαφορετικά χαρακτηριστικά, παράθυρα και στοιχεία ελέγχου, όπως κουμπιά, πλαίσια κειμένου κλπ. Επιπλέον, η χρήση GUI μπορεί να καταστήσει την ανάπτυξη των εφαρμογών πιο εύκολη και γρήγορη, καθώς παρέχουν ένα σύνολο προεπιλεγμένων στοιχείων ελέγχου που μπορεί να χρησιμοποιηθεί για να δημιουργήσει τη διεπαφή με τον χρήστη. Στην εν λόγω διπλωματική δημιουργήθηκε μηχανή επιλογής αρχείων, η οποία είναι η πρώτη γραφική διεπαφή μεταξύ του χρήστη και του προγράμματος. Αυτό καθιστά την γρήγορη και αποτελεσματική αναγνώριση των προγραμμάτων του χρήστη, προς επεξεργασία του αλγορίθμου. Έπειτα από αυτή την διαδικασία έχοντας σιγουρέψει ο χρήστης την σωστή επιλογή του αρχείου του προς προσπέλαση του αλγορίθμου, ακολουθεί η απεικόνιση πολλαπλών επιλογών ώστε να επιλεχθούν οι ανάλογοι παράμετροι στον υπολογισμό του αλγορίθμου και να υπάρξει το επιθυμητό αποτέλεσμα. Μέσω των GUI γραφικών η ενημέρωση του χρήστη, γίνεται από την εμφάνιση όλων των συντεταγμένων των αυτοκινήτων καθώς και των μοντέλων αυτών, πάνω σε έναν ψηφιακό παγκόσμιο χάρτη ο οποίος είναι λειτουργικός και εύχρηστος με τον χρήστη και του προσδίδει γρήγορη και σημαντική πληροφορία. Ένα πολύ σημαντικό εργαλείο στην επιλογή μια αγοράς του χρήστη καθώς με την απεικόνιση του χάρτη ο χρήστης έχει την δυνατότητα να αντιλαμβάνεται κυρίως εύκολα και γρήγορα την απόσταση όλων των κορυφαίων επιλογών των οποίων θα μπορούσε να κάνει, από την απόσταση στην οποία βρίσκεται.

4.3.1 Χρήση της μηχανής επιλογής αρχείων



Εικόνα 5:File Chooser

Για την αρχική επιλογή του εισαγόμενου αρχείου προς επεξεργασία και λήψη δεδομένων επιλέχθηκε η χρήση της κλάσης JFileChooser του πακέτου javax.swing. Με αυτή την κλάση διατίθεται δυνατότητα περιήγησης σε όλο τον υπολογιστή και τις συνδεδεμένες συσκευές που μπορεί να περιλαμβάνουν αρχεία για να επιλέξουμε το αρχείο που περιλαμβάνει τα δεδομένα και θα επεξεργαστούμε αργότερα για να λάβουμε από αυτά, μόνο τα χρήσιμα με βάση τις επιλογές του χρήστη. Στην Εικόνα 5:File Chooser φαίνεται η λειτουργία της JFileChooser ενώ έχουμε μεταβεί στον φάκελο skyline και έχουμε επιλέξει το αρχείο vehicles.csv ως εισαγόμενο αρχείο για επεξεργασία. Στο αρχείο (vehicles.csv) που δείχνουμε στα παραδείγματα περιλαμβάνονται δεδομένα για το Craigslist.org, που περιλαμβάνει την μεγαλύτερη συλλογή μεταχειρισμένων αυτοκίνητων στον κόσμο

4.3.2 Χρήση της επιλογής των επερωτήσεων

Εικόνα 6: Επιλογή παραμέτρων προς υπολογισμό

Για τη δημιουργία του ερωτήματος και τη ρύθμισή του με βάση τις ανάγκες μας, δημιουργήθηκε γραφικό περιβάλλον με χρήση του πακέτου javax.swing. Μετά την εισαγωγή του επιθυμητού αρχείου, διαβάζουμε την πρώτη γραμμή που αφορά τα ονόματα των πεδίων και μας δίνεται η δυνατότητα να επιλέξουμε τι αφορά το κάθε ένα και αν θέλουμε να το συμπεριλάβουμε στην εκτέλεσή μας ή όχι. Αυτές οι ρυθμίσεις αφορούν την προεπεξεργασία κα την εκτέλεση του αλγόριθμου. Συγκεκριμένα για κάθε πεδίο μπορούμε προαιρετικά να ορίσουμε:

- Display: Αν επιλεχθεί το συγκεκριμένο κουτάκι (checkbox) τότε το αντίστοιχο πεδίο θα συμπεριληφθεί στο τελικό αρχείο των αποτελεσμάτων. Με αυτό τον τρόπο ορίζουμε ότι το

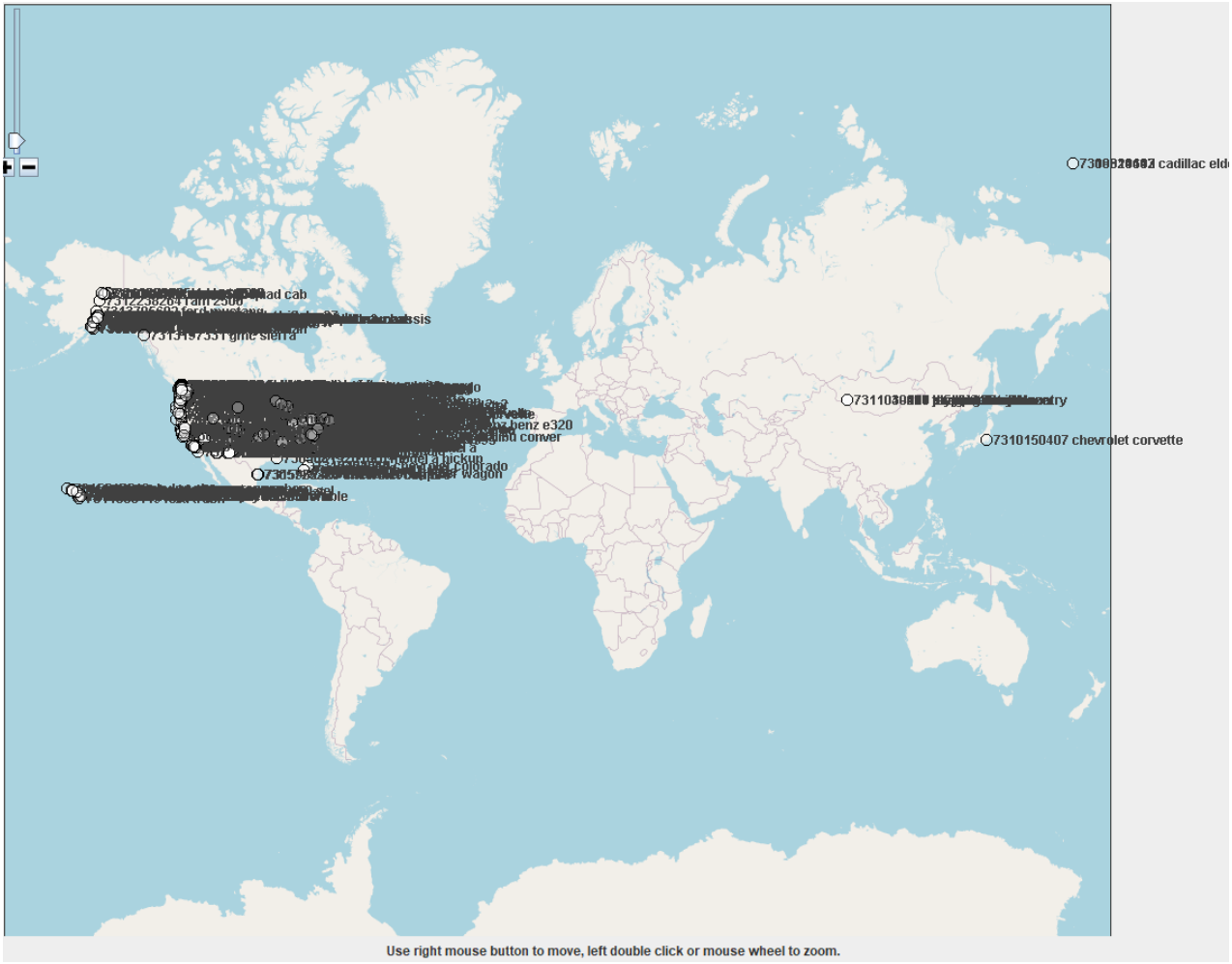
συγκεκριμένο πεδίο μας είναι χρήσιμο όταν ολοκληρωθεί η εκτέλεση παρότι δεν αφορά την ίδια την εκτέλεση ή την προεπεξεργασία.

- Calculate: Αν επιλεχθεί το συγκεκριμένο κουτάκι, τότε το αντίστοιχο πεδίο θα χρησιμοποιηθεί στην εκτέλεση του αλγόριθμου και με βάση αυτό και όποιο άλλο πεδίο επιλέξουμε θα εξαχθούν τα αποτελέσματά μας. Αυτή η ρύθμιση απαιτεί και την επιλογή του Display σε όποιο πεδίο την ενεργοποιήσουμε.
- Ascending: Εκ προεπιλογής, ο υπολογισμός των αποτελεσμάτων κατά την εκτέλεση του αλγόριθμου λαμβάνει τις μεγαλύτερες τιμές, ως τις κυρίαρχες. Επιλέγοντας το κουτάκι Ascending ορίζουμε ότι στο πεδίο στο οποίο το επιλέξαμε ο υπολογισμός θα λαμβάνει ως κυρίαρχες τιμές, τις μικρότερες τιμές. Αυτή η ρύθμιση απαιτεί και την επιλογή του Display και την επιλογή του Calculate σε όποιο πεδίο την ενεργοποιήσουμε.
- Latitude: Μπορούμε να επιλέξουμε μόνο ένα πεδίο σε αυτή τη ρύθμιση και με αυτό τον τρόπο ορίζουμε ότι το συγκεκριμένο πεδίο αποτελεί το γεωγραφικό πλάτος στο οποίο βρίσκεται η κάθε εγγραφή.
- Longitude: Μπορούμε να επιλέξουμε μόνο ένα πεδίο σε αυτή τη ρύθμιση και με αυτό τον τρόπο ορίζουμε ότι το συγκεκριμένο πεδίο αποτελεί το γεωγραφικό μήκος στο οποίο βρίσκεται η κάθε εγγραφή.

Προσοχή! Επιλέγοντας Latitude και Longitude, υποδεικνύουμε στο πρόγραμμα ότι επιθυμούμε υπολογισμό με βάση την απόσταση.

Τέλος υπάρχει το κουμπί Calculate το οποίο θα εκκινήσει την προεργασία και την εκτέλεση του αλγόριθμου. Αν έχουμε επιλέξει Latitude και Longitude, τότε θα μας ζητηθεί να εισάγουμε τις γεωγραφικές συντεταγμένες μας από τις οποίες θα υπολογιστεί η απόσταση όλων των άλλων αποστάσεων όπως περιγράφηκε προηγουμένως σε αντίστοιχη ενότητα.

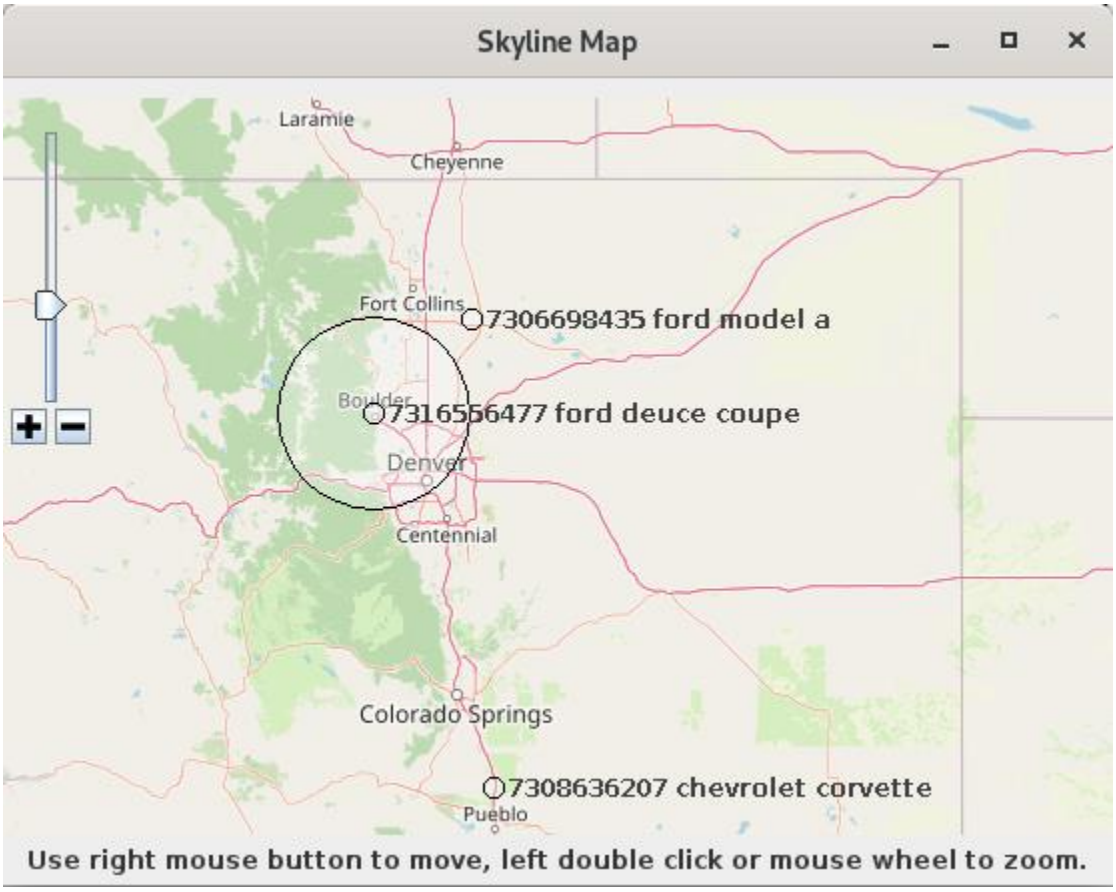
4.3.3 Διεπαφή χάρτη



Εικόνα 7: Εμφάνιση συντεταγμένων αυτοκινήτων στον χάρτη

Αν επιλέξουμε Latitude και Longitude κατά την επιλογή ρυθμίσεων για το ερώτημα, τότε έχουμε επιλέξει να γίνει και υπολογισμός με βάση την απόσταση και στο τέλος της εκτέλεσης του αλγόριθμου θα εμφανιστεί ένα γραφικό περιβάλλον προβολής χάρτη, όμοιο με του στιγμιότυπου παραπάνω το οποίο θα περιλαμβάνει τα παρακάτω:

- Παγκόσμιο χάρτη με τις ονομασίες χωρών, πολιτειών, πόλεων, χωριών, καθώς και τα σύνορα τους.
- Δυνατότητα κύλισης σε όλο το μήκος και πλάτος του κρατώντας πατημένο το δεξί κλικ του ποντικιού μας
- Δυνατότητα μεγέθυνσης και σμίκρυνσης με ροδέλα ή τα κατάλληλα κουμπιά στο αριστερό κομμάτι της προβολής
- Ένα μεγάλο κύκλο του οποίου το κέντρο αντιπροσωπεύει τις γεωγραφικές συντεταγμένες που εισήγαγε ο χρήστης κατά την τελική ρύθμιση του ερωτήματος.
- Πολλαπλούς μικρούς κύκλους οι οποίοι αντιπροσωπεύουν τις γεωγραφικές συντεταγμένες, συνοδευόμενους από τον κωδικό, τον κατασκευαστή και το μοντέλο του κάθε αποτελέσματος από τις αρχικές εγγραφές. Με βάση τον κωδικό μπορεί κανείς να ανατρέξει στο αρχείο αποτελεσμάτων και να αναζητήσει το αντίστοιχο αποτέλεσμα.



Εικόνα 8: Αντιπροσωπεύει το σημείο το οποίο επέλεξε να εμφανίσει ο χρήστης

5 *Κορυφογραμμές με απόσταση και πειραματική μελέτη*

5.1 Πειραματική ανάλυση

Στο προηγούμενο κεφάλαιο παρουσιάσαμε την εφαρμογή για τον υπολογισμό ερωτήσεων κορυφογραμμής σε χωρικά δεδομένα, καθώς και την υλοποίηση του αλγορίθμου εύρεσης κορυφογραμμών (Block Nested Loop). Σε αυτό το κεφάλαιο θα εστιάσουμε σε μια επιπλέον δυνατότητα της υλοποίησης που αφορά τον υπολογισμό ερωτήσεων κορυφογραμμής με βάση την απόσταση από ένα σημείο. Η εκτέλεση του προγράμματος ξεκινάει με την αναζήτηση του αρχείου προς επεξεργασία, καθώς και την επιλογή των παραμέτρων από τον χρήστη, ώστε να επιτυγχάνεται ο υπολογισμός με βάση τις παραμέτρους της αρέσκειας του. Αν ο χρήστης έχει επιλέξει τον υπολογισμό για τις επιλογές latitude και longtitude τότε ο κώδικας θα δημιουργήσει μία επιπλέον παράμετρο, μέσα από την οποία θα εμφανίζεται το skyline distance (s_distance). Το αποτέλεσμα αυτό θα είναι ορατό, στο αρχείο (csv) εκροής που είναι προγραμματισμένο το πρόγραμμα να δημιουργεί κατά την επιτυχή ολοκλήρωση της διαδικασίας. Σε αυτήν την περίπτωση ο υπολογισμός της ερώτησης κορυφογραμμής θα γίνεται στις παραμέτρους που δόθηκαν από τον χρήστη και επιπλέον θα λαμβάνεται ως επιπλέον διάσταση και η απόσταση από το σημείο που δόθηκε.

Πιο αναλυτικά βλέπει κανείς λοιπόν στον Πίνακα 8, ότι στην πρώτη εκτέλεση του αλγορίθμου χωρίς την παράμετρο απόσταση επιλέχθηκαν να εμφανιστούν οι παράμετροι skyline_id, id, price, year, manufacturer, model, odometer, lat, long. Από αυτές τις παραμέτρους επιλέχθηκαν να υπολογιστούν οι εμφανώς πιο ενδιαφέρον προς τον χρήστη παράμετροι, δηλαδή το price, year και odometer. Αξίζει να σημειωθεί ότι το αρχείο εκροής χωρίς την απόσταση έφερε ως αποτέλεσμα 33 εγγραφές ενώ η εκτέλεση του κώδικα με βάση την απόσταση έφερε ως αποτέλεσμα 82 εγγραφές, παρατηρώντας έτσι ότι το πλήθος των εγγραφών στα αποτελέσματα που φέρει το αρχείο εκροής με βάση την απόσταση τείνει να έχει αύξον αριθμό. Στους Πίνακες, επιλέχθηκαν προς εμφάνιση οι πρώτες 7 εγγραφές και επιπροσθέτως στον υπολογισμό των παραμέτρων προς εύρεση κορυφογραμμής, στο price και στο odometer ακολούθησε η επιλογή ascending, επηρεάζοντας τον υπολογισμό τους, δίνοντας προτεραιότητα στην ανιούσα σειρά των αποτελεσμάτων.

skyline_id	id	price	year	manufacturer	model	odometer	lat	long
235947	7310610870	2700	2022	mercedes-benz	benz e300 diesel	178000	40.8536	-74.1079
60863	7309051490	2500	2022	honda	civic cvcc	605000	37.7383	-121.4345
60864	7309047664	3500	2022	toyota	mighty max	180000	37.7383	-121.4345
292669	7314742314	4790	2022	ford	explorer eddie bauer 4x4	139700	45.497554	-122.900875
39829	7314165992	11990	2022	toyota	4runner 4wd	207834	33.694254	-117.846432
209269	7317023081	12345	2022	chevrolet	silverado	1000000	45.423	-112.1735
…	…	…	…	…	…	…	…	…

Πίνακας 6: Πειραματική εκτέλεση χωρίς την παράμετρο απόσταση

Σε αντίθεση με τον Πίνακα 18, βλέπει κανείς στον Πίνακα 19 ότι σαν πρώτο αποτέλεσμα στην έξοδο του δίνει την εγγραφή αυτή, η οποία είχε skyline_distance 0. Αν και η πρώτη εγγραφή στα αποτελέσματα του Πίνακα φανερώνεται πως στην τιμή της, υπάρχει αρκετά μεγαλύτερη διαφορά απ' ότι οι υπόλοιπες εγγραφές, ο αλγόριθμος την ξεχώρισε και την επέστρεψε στην πρώτη θέση του πίνακα καθώς ο παράγοντας απόσταση είναι 0 και του προσδίδει προτεραιότητα.

Το μηδενικό αποτέλεσμα προέκυψε καθώς το πείραμα ακολούθησε με τον εξής τρόπο. Οι συντεταγμένες της εγγραφής με id 7314165992, δηλαδή η παράμετρος lat και long εισήχθησαν πριν την εκτέλεση του πειράματος με σκοπό την παρατήρηση για την κατηγοριοποίηση του. Ο αλγόριθμος αν και έδωσε προτεραιότητα στην κορυφογραμμή της απόστασης, φαίνεται στην $2^{η}$ και $3^{η}$ εγγραφή να υπάρχουν ίδιες συντεταγμένες για δύο διαφορετικά αυτοκίνητα. Αυτό καθιστά τις δύο αυτές εγγραφές να έχουν ίδιο αποτέλεσμα απόστασης κορυφογραμμής και ο αλγόριθμος να προδίδει προτεραιότητα πρώτα στην τιμή που εκπροσωπεί το αυτοκίνητο και κατ' επέκταση στην τιμή του οδόμετρού. Αυτό παρατηρείται και στις υπόλοιπες εγγραφές καθώς μέχρι και το τέλος του πίνακα δεν υπάρχει κάποια σημαντική αλλαγή.

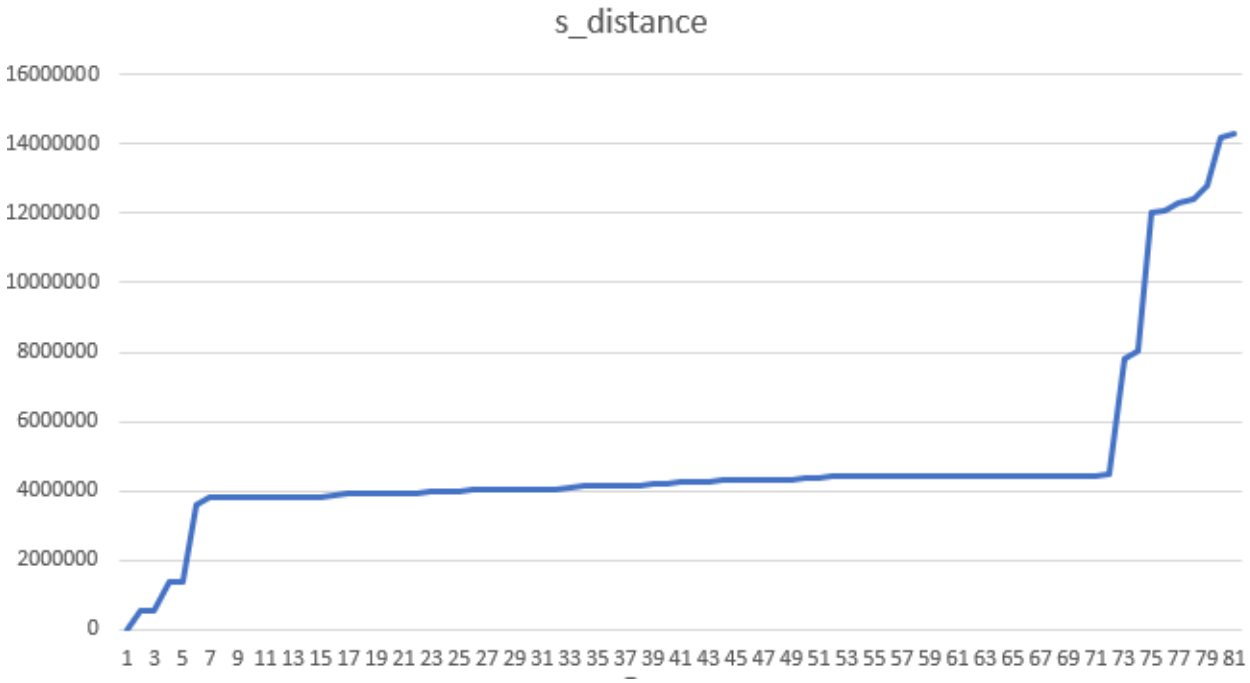
skyline_id	id	price	year	manufacturer	model	odometer	lat	long	_s_distance
39829	7314165992	11990	2022	toyota	4runner 4wd	207834	33.694254	-117.846432	0
60863	7309051490	2500	2022	honda	civic cvcc	605000	37.7383	-121.4345	554095.241
60864	7309047664	3500	2022	toyota	mighty max	180000	37.7383	-121.4345	554095.241
292669	7314742314	4790	2022	ford	explorer eddie bauer 4x4	139700	45.497554	-122.900875	1381321.957
209269	7317023081	12345	2022	chevrolet	silverado	1000000	45.423	-112.1735	1390970.611
…	…	…	…	…	…	…	…	…	…

Πίνακας 7: Πειραματική εκτέλεση με την παράμετρο απόσταση

5.2 *Συμπεράσματα*

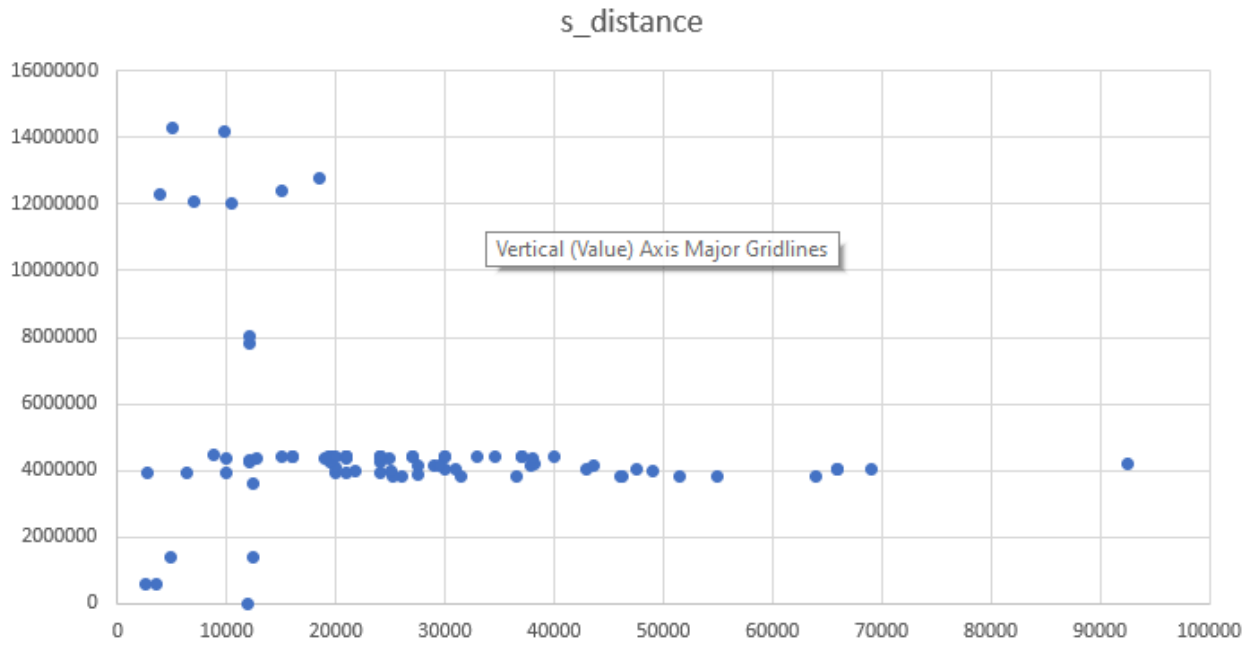
Οι κορυφογραμμές είναι μια τεχνική ανάλυσης δεδομένων που χρησιμοποιείται για να εξετάσουμε τη σχέση μεταξύ μεταβλητών. Η τεχνική αυτή μπορεί να χρησιμοποιηθεί για να αναδείξει τη σχέση μεταξύ της απόστασης και άλλων μεταβλητών.

Αν οι κορυφογραμμές δείχνουν μια θετική κλίση, αυτό σημαίνει ότι όσο αυξάνεται η μία μεταβλητή, για παράδειγμα η απόσταση, τόσο αυξάνεται και η άλλη μεταβλητή για παράδειγμα, ο χρόνος που απαιτείται για να διανύσει κάποιος αυτήν την απόσταση. Καταλαβαίνει κανείς ότι μέσα από αυτή την διαδικασία μεγάλων τιμών απόστασης και χρόνου, κατηγοριοποιούν αυτά τα αυτοκίνητα στα χαμηλά επίπεδα κορυφογραμμής, καθώς και το πλήθος αυτών είναι αρκετά μικρό. Όπως παρατηρείται και στον Πίνακα 20, ο οποίος απεικονίζει το πλήθος των κορυφογραμμών σχετικά με την απόσταση, οι πρώτες πέντε κορυφογραμμές έχουν απόσταση μικρότερη από 4000000 μέτρα, ενώ μετέπειτα ακολουθεί ένα μεγάλο πλήθος κορυφογραμμών στα σχεδόν ίδια μέτρα απόστασης. Μετά από αυτό το μεγάλο πλήθος, μόνο περίπου 10 αυτοκίνητα έχουν αρκετή διαφορά απόστασης μεταξύ τους αυξάνοντας την γραμμή εκθετικά.



Εικόνα 9: Γράφημα γραμμής σχετικά με την απόσταση

Αξίζει να σημειωθεί ότι για την περάτωση ενός επερωτήματος BNL, όσον αφορά τον χώρο και τον χρόνο, λαμβάνοντας υπόψην και τον παράγοντα απόσταση, τα πράγματα δεν επηρεάζονται σημαντικά. Αυτή την φορά στον Πίνακα 21 υπάρχει γράφημα κουκίδας απόστασης-τιμής (αυτοκινήτου) δείχνοντας επίσης και τα αυτοκίνητα αν και ίδιας απόστασης η διαφορά τιμής μεταξύ τους είναι αρκετά μεγάλη.



Εικόνα 10: Γράφημα κουκίδας σχετικά με την απόσταση, αναλογικά της τιμής

6 *Συμπεράσματα*

6.1 Σύνοψη

Στο πρώτο κεφάλαιο έγινε λεπτομερή αναφορά όσο αφορά την προεπισκόπηση της διπλωματικής, δίνοντας επεξηγήσεις για το αντικείμενο το οποίο θα μελετηθεί και θα υλοποιηθεί, καθώς επίσης και ποια θα είναι η δομή όπου θα ακολουθήσει. Πέρα από αυτό, εμβάθυνε επίσης και σε βασικές έννοιες πάνω στην μελέτη παραλλαγών ερωτήσεων κορυφογραμμής, οι οποίες είναι απαραίτητες για να κατανοήσει κάποιος την παρούσα διπλωματική διατριβή. Στο δεύτερο κεφάλαιο έλαβε θέση η έντονη ανάλυση ερωτήσεων κορυφογραμμών, αναγράφονται παραδείγματα πάνω σε αυτές, καθώς επίσης και το πως μπορεί να δημιουργηθεί μια ερώτηση κορυφογραμμών αλλά και το τι είναι μια κορυφογραμμή. Εν συνεχεία του κεφαλαίου, αναλύονται αλγόριθμοι οι οποίοι έχουν σχεδιαστεί για να μπορεί ο χρήστης να χρησιμοποιήσει ερωτήσεις κορυφογραμμών, καθώς επίσης συγκρίνονται μεταξύ τους με σκοπό την ανάδειξη των πλεονεκτημάτων τους και των μειονεκτημάτων τους. Στο τρίτο κεφάλαιο υπάρχει βιβλιογραφική ανασκόπηση με σκοπό την επαναληπτική αξιολόγηση σε θέματα των ερωτήσεων κορυφογραμμών αλλά και σε καινούργια θέματα, απόψεις και συμπεράσματα συγγραφών του αντικειμένου. Όσον αφορά το τέταρτο κεφάλαιο, γίνεται λεπτομερή επεξήγηση όλων των κλάσεων του αλγορίθμου όπου χρειάστηκαν για την υλοποίηση της εφαρμογής. Όσον αφορά την υλοποίηση του αλγορίθμου η πρώτη απόπειρα έμοιαζε πάρα πολύ με αυτή του αλγορίθμου, μίας αφελής προσέγγισης κορυφογραμμών χωρίς να υπάρχει ιδιαίτερη προτεραιότητα σε παραμέτρους και σχεδιασμένος ώστε να δίνει στην έξοδο του παραπάνω από ένα μικρό συγκεκριμένο πλήθος αποτελεσμάτων. Μετά από διαμορφώσεις, δημιουργήθηκε μία ολοκληρωμένη έκδοση του block nested loop πάνω στην οποία έγιναν βελτιώσεις όπως να εξάγει και να τα απεικονίζει τα αποτελέσματα στον χρήστη μέσα από έναν παγκόσμιο, πλήρως λειτουργικό χάρτη εφόσον έχει ληφθεί υπόψη η παράμετρος απόσταση και όχι μόνο. Στο κεφάλαιο αυτό γίνεται επίσης και επεξήγηση των γραφικών της εφαρμογής με σκοπό τη σωστή λειτουργία του από το χρήστη. Στη συνέχεια, στο πέμπτο κεφάλαιο ακολούθησε πειραματική μελέτη έχοντας για παράδειγμα προς επεξεργασία ένα αρχείο με μία μεγάλη ποικιλία αυτοκινήτων. Έπειτα περιγράφονται δύο πετυχημένες εκτελέσεις από τις οποίες η μία εκτέλεση ήταν χωρίς την παράμετρο της απόστασης ενώ η αμέσως επόμενη ήταν με την παράμετρο. Οι δύο αυτές εκτελέσεις αναλύθηκαν και συγκρίθηκαν μεταξύ τους δίνοντας στην έξοδο τους σωστά αποτελέσματα και τέλος έφεραν στην επιφάνεια σημαντικά συμπεράσματα.

Εν κατακλείδι, η μελέτη και οι δυσκολίες που αντιμετωπίστηκαν στην παρούσα διπλωματική εργασία οδήγησαν στην δημιουργία ενός αλγόριθμου μέσα από τον οποίο επιτυγχάνεται η απλοποίηση στην λήψη δύσκολων αποφάσεων σε ένα μεγάλο σύνολο δεδομένων παρέχοντας σαφή αναπαράσταση επιλογών για τον χρήστη, έχοντας ως κριτήριο βάσης την απόσταση. Η όλη διαδικασία είναι προγραμματισμένη να γίνεται σε πραγματικό χρόνο, καθώς επίσης λαμβάνονται υπόψη μόνο τα κυρίαρχα σημεία.

6.2 Επέκταση

Οι παραλλαγές ερωτήσεων κορυφογραμμής με βάση την απόσταση είναι ένας πολύ ενδιαφέρον τομέας για έρευνα, και βέβαια εξελίσσεται διαρκώς. Σίγουρα, κάποιες πιθανές κατευθύνσεις που θα μπορέσει να ακολουθήσει στο μέλλον για την εξέλιξη του είναι η ανάπτυξη των ερωτήσεων όπου αφορούν γενικώς την τοπολογία και τα δίκτυα, όμως πιο εξειδικευμένα μπορούν να γίνουν τα εξής:

- Ανάπτυξη περισσότερων μεθόδων επίλυσης: Υπάρχει η δυνατότητα να αναπτυχθούν νέες τεχνικές και αλγόριθμοι για να μπορούν να απαντηθούν περισσότερες αλλαγές ερωτήσεων κορυφογραμμών. Οι τεχνικές αυτές θα περιλαμβάνουν προηγμένη ανάπτυξη όσον αφορά αλγορίθμους αναζήτησης, την χρήση μηχανικής μάθησης ή ακόμα και την εφαρμογή προηγμένων μεθόδων ανάλυσης γράφων.
- Αύξηση του όγκου και της ποικιλίας διαθέσιμων παραλλαγών: Θα μπορούσε να ενισχυθεί το φάσμα με το πλήθος διαθέσιμων παραλλαγών ερωτήσεων κορυφογραμμών με σκοπό να παρέχονται πιο ακριβείς και πληρέστερες απαντήσεις. Αυτό βέβαια περιλαμβάνει την ανάλυση της συντακτικής δομής των ερωτήσεων, καθώς επίσης και την χρήση σημασιολογικών και συντακτικών εργαλείων αλλά και τέλος, την βελτιστοποίηση της αλγοριθμικής επεξεργασίας των ερωτήσεων.
- Εξερεύνηση και ανακάλυψη διαφορετικών μετρήσεων απόστασης: Η μελέτη θα μπορούσε να στραφεί και στην εξέταση διαφορετικών μετρήσεων απόστασης όπως είναι η απόσταση Levenshtein, η απόσταση Hamming, καθώς μέσα από αυτές τις διαφορετικές τεχνικές αλλάζει η μεθοδολογία στην δημιουργία σημειακών χωρικών δεδομένων και στην διεκπεραίωση των συστημάτων ερωτήσεων κορυφογραμμών.
- Δημιουργία νέων προκλήσεων: Σίγουρα η εξέλιξη συμβαίνει έχοντας ως σκοπό ο άνθρωπος να αντιμετωπίσει την δυσκολία που τον διακατέχει. Με την δημιουργία νέων προκλήσεων θα υπάρχει ένα κίνητρο προς επίλυση του προβλήματος μέσα από την οποία διαδικασία θα ακολουθήσει μία πετυχημένη εξέλιξη. Αυτό φυσικά δεν συμβαίνει μόνο στον τομέα των ερωτήσεων κορυφογραμμών καθώς το νόημα του καλύπτει ένα μεγάλο φάσμα ειδικά στην ζωή του ανθρώπου.
- Συσχέτιση με άλλους κλάδους επιστήμης: Η μελέτη παραλλαγών ερωτήσεων κορυφογραμμών με βάση την απόσταση μπορεί να εφαρμοστεί για παράδειγμα στους τομείς της βιολογίας και της χημείας. Ένα πολύ καλό παράδειγμα μέσα από το οποίο μπορεί να καταλάβει κανείς την χρησιμότητα είναι η κατάταξη δομών μορίων. Στο κλάδο της χημείας μπορεί να χρησιμοποιηθεί η απόσταση μεταξύ κυρίαρχων σημείων για να καταταχτούν δομές μορίων σε μία βάση δεδομένων. Αντίστοιχα θα μπορούσε η διαδικασία αυτή να λάβει δράσει στον τομέα της βιολογίας για την εξερεύνηση δομών πρωτεϊνών.

7 *Βιβλιογραφία*

Bartolini, I., Ciaccia, P., & Patella, P. (2008, Νοέμβριος). Efficient sort-based skyline evaluation. *ACM Trans. Database Syst. 33, 4, Article 31*.

Borzsony, S., Kossmann, D., & Stocker, K. (2001). The Skyline operator. *Proceedings 17th International Conference on Data Engineering*, σσ. 421-430.

Borzsony, S., Kossmann, D., & Stocker, K. (2005). Skyline queries against mobile lightweight devices. *Proceedings of the 21st International Conference on Data Engineering*.

Ferhatosmanoglu, H., Stanoi, I., Agrawal, D., & Abbadi, A. (2001). Constrained Nearest Neighbor Queries. *SSTD*, σσ. 257-276.

Guttman, A. (1984). R Trees: A Dynamic Index Structure for Spatial Searching. *MOD'84, Proceedings of Annual Meeting, Boston, Massachusetts*.

Hjaltason, G., & Samet, H. (1999). Distance browsing in spatial databases. *ACM Transactions on Database Systems (TODS), 24(2)*, σσ. 265-318.

Kalyvas, C., & Tzouramanis, T. (2017). A Survey of Skyline Query Processing. *arXiv preprint arXiv:1704.01788*.

Kossmann, D., Ramsak, F., & Rost, S. (2002). Shooting Stars in the Sky: An Online Algorithm for Skyline Queries. *Proceedings of the 28th International Conference on Very Large Databases*, σσ. 275-286.

Kung, H., Luccio, F., & Preparata, F. (1975). On finding the maxima of a set of vectors. *Journal of the ACM (JACM) 22(4)*, σσ. 469-476.

Lai, R., Lee, D., & Yin, J. (2012). Efficient computation of the skyline cube. *Proceedings of the 28th International Conference on Data Engineering*.

Papadias, D., Tao, Y., Fu, G., & Seeger, B. (2003). An optimal and progressive algorithm for Skyline Queries. *ACM SIGMOD, San Diego, CA, USA*.

Preparata, F., & Shamos, M. (1985). Computational Geometry: An Introduction. *Springer-Verlag*.

Roussopoulos, N., Kelley, S., & Vincent, F. (1995, Μάιος). Nearest neighbor queries. *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, σσ. 71-79.

Sharifzadeh, M., & Shahabi, C. (2006, Σεπτέμβριος). The Spatial Skyline Queries. *Proceedings of the 32nd international conference on Very large data bases*, σσ. 751-762.

Stojmenović, I., & Miyakawa, M. (1988). An optimal parallel algorithm for solving the maximal elements problem in the plane. *Parallel Computing, 7(2)*, σσ. 249-251.

Wu , T., & Chen, M. (2005). Top-k dominating queries. *Proceedings of the 13th annual ACM international workshop on Geographic information systems*.

Zheng, B., Lu, Y., Wang, W., & Zhou, X. (2018). Processing dynamic skyline queries in outsourced cloud computing. *Proceedings of the 34th International Conference on Data Engineering*.