ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«ΒΙΒΛΙΟΘΗΚΗ PYTHON ΓΙΑ SKYLINE OPERATOR»

Του φοιτητή
Εμμανουήλ Παντούση
Αρ. Μητρώου: 175103

Επιβλέπων
Αντώνης Σιδηρόπουλος
Επίκουρος Καθηγητής

14-06-2021

Τίτλος Π.Ε: Βιβλιοθήκη Python για Skyline Operator.
Κωδικός Π.Ε: 20196
Ονοματεπώνυμο φοιτητή: Εμμανουήλ Παντούσης.
Ονοματεπώνυμο εισηγητή: Αντώνης Σιδηρόπουλος
Ημερομηνία ανάληψης Π.Ε. 14-10-2020
Ημερομηνία περάτωσης Π.Ε. 14-06-2021

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

«Στην οικογένειά μου»

Πρόλογος

Ολοκληρώνοντας τον κύκλο των σπουδών μου στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, κλήθηκα να επιλέξω ένα θέμα για την εκπόνηση της πτυχιακής μου εργασίας. Μη έχοντας κάτι συγκεκριμένο στο μυαλό μου, επέλεξα το παρόν θέμα καθώς μου φάνηκε ενδιαφέρον τόσο από αλγοριθμικής άποψης όσο και λόγω της γλώσσας προγραμματισμού. Το όφελος από την εκπόνηση της παρούσας εργασίας ήταν μεγάλο καθώς πήρα σημαντικές γνώσεις σχετικά με τον τρόπο προσέγγισης του σχεδιασμού ενός αλγορίθμου αλλά και τους τρόπους βελτίωσης της απόδοσής του κατά την κλιμάκωση ενός προβλήματος. Επιπλέον, διεύρυνα τις γνώσεις μου πάνω στη γλώσσα προγραμματισμού python.

Περίληψη

Ο τελεστής skyline αποτελεί βασικό εργαλείο στην επίλυση προβλημάτων βελτιστοποίησης όπου απαιτείται η λήψη αποφάσεων με βάση πολλαπλά κριτήρια, καθώς εξάγει τα πιο ενδιαφέροντα στοιχειά μέσα από ένα πολυδιάστατο σύνολο δεδομένων. Τα τελευταία χρόνια, το ευρύ πεδίο εφαρμογής του τον έχει καταστήσει πολύ δημοφιλή. Παράλληλα, τα διαθέσιμα δεδομένα στο διαδίκτυο αυξάνονται συνεχώς με εκθετικό ρυθμό. Όμως, όσο ο αριθμός των δεδομένων αυξάνεται, το υπολογιστικό κόστος για τη σύγκρισή τους γίνεται ολοένα και μεγαλύτερο καθιστώντας τον υπολογισμό του skyline σε πολυδιάστατα δεδομένα μια μεγάλη πρόκληση. Επιπλέον, οι ελλιπείς τιμές που παρουσιάζονται σε ορισμένα σύνολα δεδομένων, συνιστούν ένα ακόμα εμπόδιο στον αποδοτικό υπολογισμό του. Στην παρούσα πτυχιακή εργασία υλοποιείται μια βιβλιοθήκη python για τον υπολογισμό όλων των επιπέδων skyline ενός συνόλου δεδομένων και ελέγχεται πειραματικά η απόδοσή της σε πολλαπλά κλιμακούμενα σύνολα δεδομένων. Τα πειραματικά αποτελέσματα είναι τα αναμενόμενα με βάση τη θεωρία.

Python library for Skyline Operator

Emmanouil Pantousis

Abstract

The skyline operator is a common operator in optimization problem solving where multi-criteria decision making is required, as it extracts the most interesting elements from a multidimensional dataset. Recently, the wide application field of skyline operator has increased its popularity. Besides, the available data on the Internet is constantly increasing. However, more data results to higher computational cost, thus calculating the skyline of such datasets is challenging. Furthermore, incomplete data makes the calculation even more challenging. This thesis proposes a python library for the calculation of all skyline levels of a given dataset. Its performance is tested on multiple datasets of various lengths and dimensions. The results of the performance tests are as expected.

Ευχαριστίες

Ευχαριστώ ιδιαιτέρως τον εισηγητή και επιβλέποντα κ. Αντώνη Σιδηρόπουλο που μου εμπιστεύτηκε την εκπόνηση της παρούσας πτυχιακής εργασίας, αλλά και για την άρτια συνεργασία και καθοδήγησή του που κατέστησε εφικτή την ολοκλήρωσή της.

Περιεχόμενα

Πρόλογος v
Περίληψη vi
Abstract vii
Ευχαριστίες viii
Περιεχόμενα ix
Κατάλογος Σχημάτων xi
Κατάλογος Πινάκων xi
Κατάλογος Εικόνων xi
Συντομογραφίες xii
Κεφάλαιο 1ο: Εισαγωγή 13
1.1 Περιγραφή του προβλήματος 13
1.2 Στόχος της εργασίας 13
1.3 Οργάνωση της εργασίας 13
Κεφάλαιο 2ο: Ερωτήματα Skyline 15
2.1 Εισαγωγή 15
2.2 Βασικές έννοιες 15
2.2.1 Κυριαρχία 15
2.2.2 Skyline 16
2.2.3 Η Κατάρα των Διαστάσεων 18
2.3 Παρόμοια προβλήματα 19
2.3.1 Convex Hull 19
2.3.2 Maximal Vectors 19
2.3.3 Pareto Front 19
2.4 Εφαρμογές 20
2.5 Επεκτάσεις 21
2.5.1 Top-k Κυρίαρχα Ερωτήματα 21
2.5.2 Δυναμικό Skyline 22
2.5.3 K-κυρίαρχο Skyline 22
2.5.4 Skycube 22
2.5.5 Πιθανοτικό Skyline 23
2.5.6 Αντίστροφο Skyline 24
2.5.7 Ομαδικό Skyline 24

2.5.8 Χωρικό Skyline 24
Κεφάλαιο 3ο: Αλγόριθμοι για την εύρεση skyline 26
3.1 Εισαγωγή 26
3.2 Block Nested Loops 26
3.3 Divide and Conquer 27
3.4 Bitmap 28
3.5 Index 30
3.6 Nearest Neighbor 32
3.7 Branch and Bound Skyline 33
3.8 Sort Filter Skyline 35
3.9 Linear Elimination Sort for Skyline 36
3.10 Sort and Limit Skyline Algorithm 37
3.11 Άλλοι Αλγόριθμοι 37
Κεφάλαιο 4ο: Υλοποίηση τελεστή skyline στη γλώσσα Python 39
4.1 Η γλώσσα Python 39
4.2 Δομοστοιχεία, πακέτα και βιβλιοθήκες 39
4.3 Δημιουργία και δημοσίευση βιβλιοθήκης Python 39
4.4 Βελτιστοποίηση κώδικα Python για επιστημονικές εφαρμογές 39
4.5 Ανάπτυξη βιβλιοθήκης Python για υπολογισμό skyline 40
Κεφάλαιο 5ο: Πειραματική αξιολόγηση 42
5.1 Περιβάλλον εκτέλεσης 42
5.2 Πειραματικά σύνολα δεδομένων 42
5.3 Αποτελέσματα εκτέλεσης 42
Κεφάλαιο 6ο: Συμπεράσματα και προτάσεις για βελτίωση 44
ΒΙΒΛΙΟΓΡΑΦΙΑ 45

Κατάλογος Σχημάτων

Σχήμα 2.1: Γραφική απεικόνιση της κυριαρχίας..... 16
Σχήμα 2.2: Skyline για μεγιστοποίηση και των δύο παραμέτρων 17
Σχήμα 2.3: Skyline για μεγιστοποίηση της παραμέτρου x και ελαχιστοποίηση της παραμέτρου y 17
Σχήμα 2.4: Skyline για ελαχιστοποίηση της παραμέτρου x και μεγιστοποίηση της παραμέτρου y 18
Σχήμα 2.5: Ποσοστό skyline σε συνθετικά σύνολα δεδομένων..... 18
Σχήμα 2.6: Ελάχιστο κυρτό πολύγωνο (ενωμένα σημεία) σε σχέση με όλα τα skylines (μαυρισμένα σημεία) 19
Σχήμα 2.7: Αποτέλεσμα top-4 κυρίαρχου ερωτήματος..... 22
Σχήμα 3.1: Διαίρεση του συνόλου δεδομένων από τον αλγόριθμο DC..... 28
Σχήμα 3.2: Γραφική απεικόνιση του αλγορίθμου NN 33
Σχήμα 3.3: Ελάχιστα Περικλείοντα Ορθογώνια ενός συνόλου δεδομένων..... 34
Σχήμα 3.4: R-δέντρο για τα δεδομένα του σχήματος 3.3..... 34

Κατάλογος Πινάκων

Πίνακας 2.1: Σύνολο δεδομένων S..... 23
Πίνακας 2.2: Επιμέρους skylines για όλα τα υποσύνολα των διαστάσεων του S..... 23
Πίνακας 3.1: Δυαδική αναπαράσταση δεδομένων και bit-slices 29
Πίνακας 3.2: Οι λίστες που δημιουργεί ο αλγόριθμος Index 31
Πίνακας 3.3: Διάσχιση του R-δέντρου και περιεχόμενα του σωρού και των skylines σε κάθε βήμα .. 35
Πίνακας 3.4: Κανονικοποιημένη μορφή και εντροπία ενός συνόλου δεδομένων..... 36
Πίνακας 5.1: Χρόνος εκτέλεσης του αλγορίθμου σε δευτερόλεπτα για συνθετικά σύνολα δεδομένων ..... .42
Πίνακας 5.2: Χρόνος εκτέλεσης του αλγορίθμου σε δευτερόλεπτα για πραγματικά σύνολα δεδομένων 43

Κατάλογος Διαγραμμάτων

Διάγραμμα 5.1: Χρόνος εκτέλεσης του αλγορίθμου σε δευτερόλεπτα για συνθετικά σύνολα δεδομένων……43

Κατάλογος Εικόνων

Εικόνα 4.1: Εκτέλεση κύριας συνάρτησης της βιβλιοθήκης..... 40

Συντομογραφίες

Π.Ε. Πτυχιακή Εργασία

Κεφάλαιο 1ο: Εισαγωγή

1.1 Περιγραφή του προβλήματος

Δεν είναι λίγες οι φορές που κάποιος θέλει να αναζητήσει μέσα από ένα σύνολο δεδομένων το καλύτερο στοιχείο ή την ομάδα στοιχειών που ταιριάζει καλύτερα στις απαιτήσεις ή τις ανάγκες του. Αυτή η αναζήτηση συνήθως στοχεύει στην επίτευξη του βέλτιστου συμβιβασμού ανάμεσα σε ένα πλήθος από διαφορετικά κριτήρια.

Για παράδειγμα, σε ένα σύνολο δεδομένων από ξενοδοχειακές μονάδες, κάθε ξενοδοχείο μπορεί να διαθέτει δύο αριθμητικά χαρακτηριστικά όπως το κόστος διαμονής και η απόσταση από την παραλία. Σε αυτή την περίπτωση, ο χρήστης ενδιαφέρεται για το ξενοδοχείο με το μικρότερο κόστος και τη μικρότερη απόσταση από την παραλία. Επειδή, όμως, τα κριτήρια αυτά είναι αντικρουόμενα και το πιθανότερο είναι να μην ικανοποιούνται ταυτόχρονα από το ίδιο ξενοδοχείο, γεννάται η ανάγκη για ένα είδος αναζήτησης που θα επιστρέφει ένα σύνολο από «ενδιαφέροντα» - με βάση τα κριτήρια που έχει θέσει ο χρήστης - ξενοδοχεία.

Οι γλώσσες ερωτοαποκρίσεων, όπως η SQL, δεν είναι σε θέση να ανταποκριθούν σε αυτό το είδος αναζήτησης. Η μόνη δυνατότητα που παρέχουν για την αναζήτηση των καλύτερων στοιχείων από ένα σύνολο δεδομένων είναι η παραδοσιακή αύξουσα ή φθίνουσα κατάταξη με τον τελεστή ORDER BY και οι τελεστές MIN και MAX για την εύρεση της ελάχιστης και μέγιστης τιμής ενός χαρακτηριστικού αντίστοιχα. Το παραπάνω, αρκεί για έναν χρήστη που αναζητεί την καλύτερη τιμή ενός χαρακτηριστικού, όχι όμως και για πιο σύνθετες αναζητήσεις που περιλαμβάνουν περισσότερα του ενός χαρακτηριστικά. Με τον τελεστή skyline γίνεται δυνατή η αναζήτηση των καλύτερων στοιχείων βάση οποιουδήποτε αριθμού χαρακτηριστικών τους.

Από τα παραπάνω, γίνεται αντιληπτή η αξία του τελεστή skyline ως προς τη λήψη πολυκριτηριακών αποφάσεων πάνω σε πολυδιάστατα δεδομένα.

1.2 Στόχος της εργασίας

Στόχος της παρούσας Π.Ε. είναι η υλοποίηση και η πειραματική εφαρμογή μιας βιβλιοθήκης υπολογισμού skylines στη γλώσσα Python μετά από μελέτη και κατανόηση του τελεστή skyline. Επιπλέον, η βιβλιοθήκη αξιολογείται πειραματικά σε συνθετικά και πραγματικά σύνολα δεδομένων κλιμακούμενου μεγέθους. Τέλος, συγκρίνονται και συζητούνται τα αποτελέσματα της πειραματικής αξιολόγησης του υλοποιημένου αλγορίθμου και προτείνονται μελλοντικές βελτιώσεις.

1.3 Οργάνωση της εργασίας

Στο Κεφάλαιο 2 γίνεται παρουσίαση του θεωρητικού υπόβαθρου του προβλήματος. Συγκεκριμένα, γίνεται εισαγωγή στις βασικές έννοιες που σχετίζονται με το κύριο πρόβλημα ενώ επιπλέον γίνεται αναφορά σε παρόμοια προβλήματα, παρουσιάζονται εφαρμογές αλλά και πλήθος επεκτάσεων.

Στο Κεφάλαιο 3 αναλύονται οι κυριότεροι αλγόριθμοι που έχουν αναπτυχθεί για την επίλυση του προβλήματος, παρουσιάζονται τα πλεονεκτήματα και μειονεκτήματά τους καθώς και η πολυπλοκότητά τους.

Στο Κεφάλαιο 4 γίνεται επισκόπηση της γλώσσας Python και των εξωτερικών βιβλιοθηκών που χρησιμοποιήθηκαν για την υλοποίηση της παρούσας βιβλιοθήκης όπως επίσης και της διαδικασίας που απαιτείται για την προσθήκη της στο επίσημο αποθετήριο της Python.

Στο Κεφάλαιο 5 παρουσιάζονται τα αποτελέσματα της πειραματικής αξιολόγησης της βιβλιοθήκης σε συνθετικά και πραγματικά σύνολα δεδομένων κλιμακούμενου μεγέθους και διαστάσεων.

Στο Κεφάλαιο 6 παρατίθενται τα συμπεράσματα που προκύπτουν από τα πειραματικά αποτελέσματα και προτείνονται μελλοντικές βελτιώσεις.

Κεφάλαιο 2ο: Ερωτήματα Skyline

2.1 Εισαγωγή

Το 2001, οι Borzsony et al. [1] πρότειναν την επέκταση της SQL με ένα νέο τελεστή που ονόμασαν τελεστή skyline (skyline operator), η σύνταξη του οποίου φαίνεται παρακάτω:

SELECT ... FROM ... WHERE ...

GROUP BY ... HAVING ...

SKYLINE OF [DISTINCT] d_1 [MIN | MAX | DIFF], ..., d_m [MIN | MAX | DIFF]

ORDER BY ...

Όπου $d_1 \dots d_m$ είναι τα χαρακτηριστικά τα οποία παράλληλα δηλώνουν τις διαστάσεις του skyline και *min, max, diff* είναι τελεστές που δηλώνουν αν επιθυμείται ελαχιστοποίηση, μεγιστοποίηση ή απλά διαφορετική τιμή για το εκάστοτε χαρακτηριστικό. Ο προαιρετικός τελεστής *distinct* αφορά στη διαχείριση των διπλότυπων εγγραφών.

Με βάση τα παραπάνω, στο παράδειγμα με τα ξενοδοχεία που αναφέρθηκε στην Εισαγωγή, επιθυμείται ελαχιστοποίηση τόσο στο χαρακτηριστικό του κόστους διαμονής όσο και στο χαρακτηριστικό της απόστασης από την παραλία.

2.2 Βασικές έννοιες

Η χρήση του τελεστή skyline επιστρέφει τις ενδιαφέρουσες πλειάδες στοιχειών. Ως ενδιαφέρουσες πλειάδες νοούνται όλες οι πλειάδες που δεν είναι χειρότερες - βάσει των κριτηρίων που έχουν τεθεί - από κάθε άλλη πλειάδα σε όλες τις διαστάσεις. Πιο αυστηρά, ως skyline ορίζεται το σύνολο των στοιχείων που δεν «κυριαρχούνται» από κανένα άλλο στοιχείο. Ένα στοιχείο κυριαρχεί έναντι ενός άλλου στοιχείου αν είναι τουλάχιστον όσο καλό σε όλες τις διαστάσεις και αυστηρά καλύτερο σε τουλάχιστον μία διάσταση.

Η εφαρμογή μιας συνάρτησης βαθμολόγησης, όπως μια γραμμική συνάρτηση σταθμισμένου αθροίσματος, στο σύνολο των δεδομένων, μπορεί να επιστρέψει αυτά τα στοιχεία και μάλιστα με φθίνουσα σειρά ενδιαφέροντος. Τις περισσότερες φορές, όμως, η στάθμιση των επιμέρους χαρακτηριστικών δεν είναι εκ των προτέρων γνωστή. Τα στοιχεία που επιστρέφει ο τελεστής skyline είναι πρακτικά όλα τα στοιχεία όλων των γραμμικών συναρτήσεων.

2.2.1 Κυριαρχία

Δεδομένου ενός d-διάστατου συνόλου δεδομένων το οποίο αποτελείται από n d-διάστατα στοιχεία, ένα στοιχείο $p = \{p_1, p_2, \dots, p_d\}$ ανήκει στο σύνολο δεδομένων και p_i είναι η τιμή του στη διάσταση i. Για οποιαδήποτε δύο στοιχεία p και q, το p καλείται κυρίαρχο έναντι του q και συμβολίζεται ως $p \prec q$ αν και μόνο αν [2]:

- Για κάθε $i \in \{1, 2, \dots, d\}$, τότε $p_i \leq q_i$.
- Υπάρχει τουλάχιστον ένα $j \in \{1, 2, \dots, d\}$ τέτοιο ώστε $p_i < q_i$.

Αν ούτε $p \prec q$ αλλά ούτε και $q \prec p$, τότε τα στοιχεία καλούνται μη συγκρίσιμα και συμβολίζονται ως $p \diamond q$.

Εκτός του παραπάνω ορισμού, η κυριαρχία μπορεί να οριστεί και πιο αυστηρά αν για οποιαδήποτε δύο στοιχεία p και q, το p καλείται κυρίαρχο έναντι του q αν και μόνο αν:

- Για κάθε $i \in \{1, 2, \dots, d\}$, τότε $p_i < q_i$.

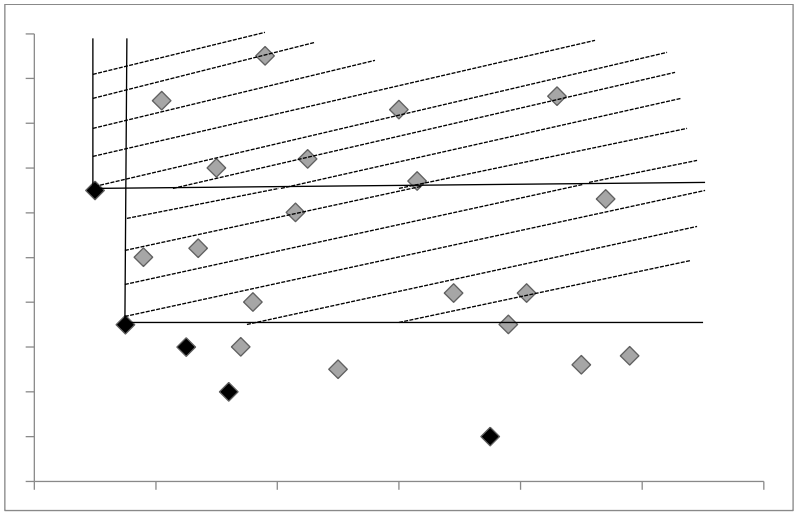
Οι παραπάνω ορισμοί αφορούν προβλήματα ελαχιστοποίησης, δηλαδή προβλήματα στα οποία επιδιώκεται ελαχιστοποίηση όλων των παραμέτρων. Γραφική απεικόνιση της έννοιας της κυριαρχίας για ένα πρόβλημα ελαχιστοποίησης σε σύνολο δεδομένων δύο διαστάσεων, παρουσιάζεται στο Σχήμα 2.1. Ενδεικτικά, είναι γραμμοσκιασμένες οι περιοχές κυριαρχίας δύο εκ των πέντε κυρίαρχων στοιχείων.

Για την κυριαρχία ισχύει η μεταβατική ιδιότητα, δηλαδή $p \prec q \wedge q \prec r \Rightarrow p \prec r$, η οποία όμως δεν ισχύει και για τα μη συγκρίσιμα στοιχεία.

Αντίθετα, για τα μη συγκρίσιμα στοιχεία ισχύει η ανακλαστικότητα, δηλαδή $p \diamond p$ η οποία όμως δεν ισχύει για την κυριαρχία, δηλαδή $p \not\prec p$.

Επιπλέον, για την κυριαρχία ισχύει η ασυμμετρία, δηλαδή $p \succ q \Rightarrow q \not\prec p$.

Τέλος, αποδεικνύεται ότι για οποιαδήποτε μονότονη συνάρτηση βαθμολόγησης $f : T \rightarrow \mathbb{R}$ και οποιοδήποτε σύνολο στοιχείων M, το skyline του M πάντα θα περιέχει το στοιχείο $p \in M$ που μεγιστοποιεί αυτή τη συνάρτηση.



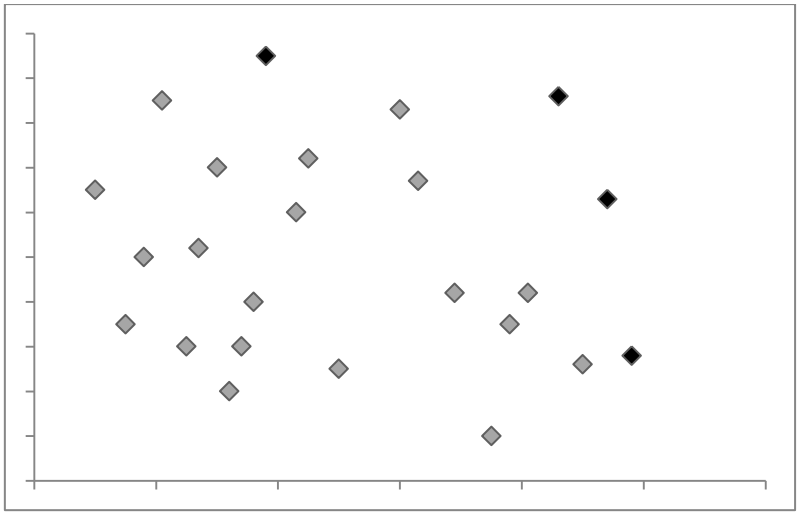
Σχήμα 2.1: Γραφική απεικόνιση της κυριαρχίας.

2.2.2 Skyline

Δεδομένου ενός συνόλου δεδομένων D, ένα σημείο $p \in D$ ανήκει στο skyline του D αν και μόνο αν δεν υπάρχει κανένα άλλο σημείο $q \in D$ που να κυριαρχεί έναντι του p [2]. Το σύνολο των στοιχείων που δεν κυριαρχούνται από τα υπόλοιπα στοιχεία, καλείται skyline. Το όνομα αυτό ουσιαστικά αντανακλά τη γραφική απεικόνιση των κυρίαρχων στοιχειών ενός συνόλου στο επίπεδο, καθώς αυτά

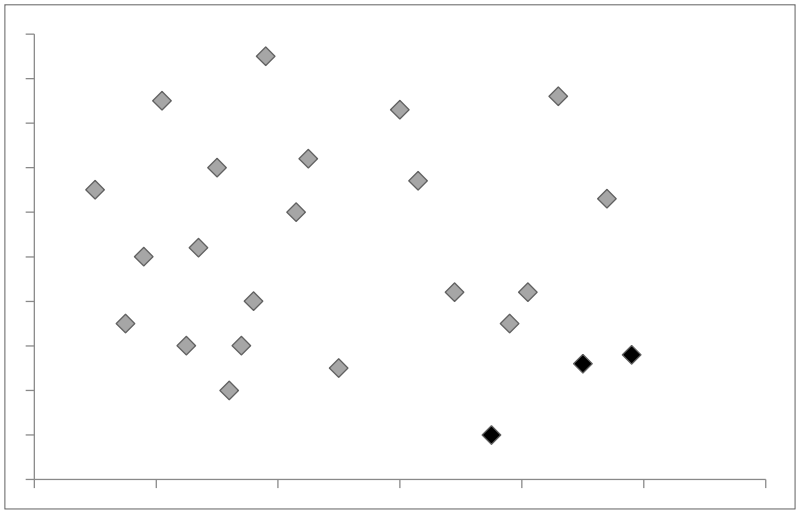
σχηματίζουν διαδοχικές κορυφές όπως οι κορυφές μιας οροσειράς ή των κτιρίων μιας πόλης που αντικρίζει κάποιος κοιτάζοντας τον ορίζοντα.

Σε αντιπαραβολή με το Σχήμα 2.1, όπου απεικονίζεται το skyline για ελαχιστοποίηση και των δύο παραμέτρων, στο Σχήμα 2.2 απεικονίζεται το skyline για μεγιστοποίηση και των δύο παραμέτρων του ίδιου συνόλου δεδομένων.

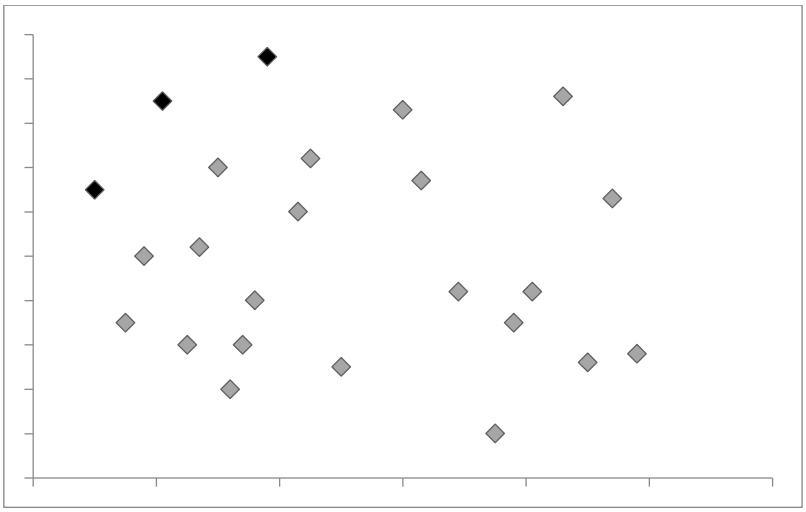


Σχήμα 2.2: Skyline για μεγιστοποίηση και των δύο παραμέτρων.

Αντίστοιχα, στα σχήματα 2.3 και 2.4 απεικονίζονται τα skylines για μεγιστοποίηση του ενός χαρακτηριστικού και ελαχιστοποίηση του άλλου.



Σχήμα 2.3: Skyline για μεγιστοποίηση της παραμέτρου *x* και ελαχιστοποίηση της παραμέτρου *y*.



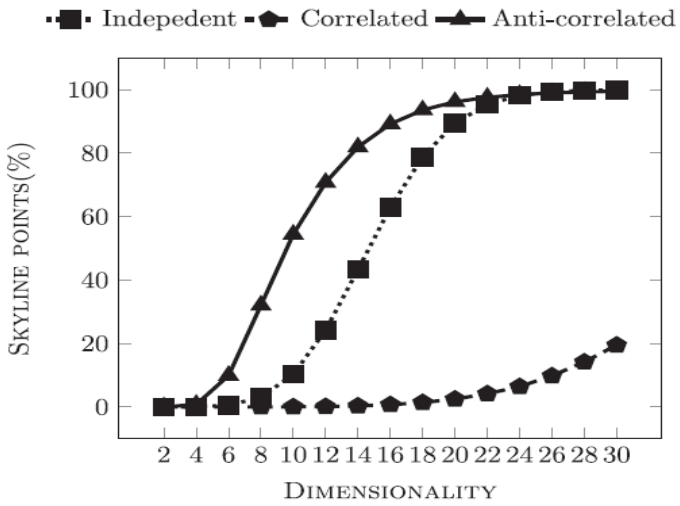
Σχήμα 2.4: Skyline για ελαχιστοποίηση της παραμέτρου *x* και μεγιστοποίηση της παραμέτρου *y*.

2.2.3 Η Κατάρα των Διαστάσεων

Από τον ορισμό της κυριαρχίας γίνεται εύκολα αντιληπτό ότι όσο περισσότερες είναι οι διαστάσεις που συγκρίνονται, τόσο μικρότερη είναι η πιθανότητα για ένα στοιχείο να κυριαρχεί έναντι ενός άλλου. Το μέγεθος του skyline αυξάνεται καθώς αυξάνεται ο αριθμός των διαστάσεων, φαινόμενο γνωστό και ως «κατάρα των διαστάσεων» (Curse of Dimensionality). Όμως, όσο μεγαλύτερο είναι το skyline τόσο περισσότερο χάνει την αξία του αφού δεν μπορεί να προσφέρει ουσιαστική πληροφορία.

Δεδομένου ενός ομοιόμορφου και ανεξάρτητου συνόλου δεδομένων μεγέθους *n* με τιμές από τον *d*-διάστατο υπερκύβο $[0,1]^d$, το αναμενόμενο μέγεθος του skyline είναι ασυμπτωτικό στο $\frac{(\log n)^{d-1}}{(d-1)!}$ για μεγάλο *n* και πεπερασμένο *d*. [3]

Όπως φαίνεται στο Σχήμα 2.5, το ποσοστό των στοιχείων του skyline επί του συνόλου των δεδομένων αυξάνεται απότομα με την αύξηση των διαστάσεων για σύνολα δεδομένων με αντίστροφη συσχέτιση, λιγότερο απότομα για ανεξάρτητα σύνολα δεδομένων, ενώ για συσχετισμένα σύνολα δεδομένων η αύξηση γίνεται αισθητή μόνο σε πολύ μεγάλο αριθμό διαστάσεων.



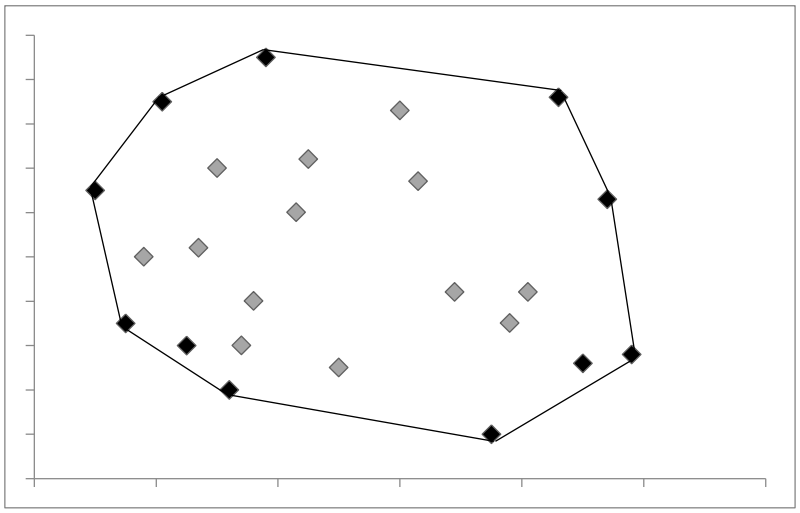
Σχήμα 2.5: Ποσοστό του skyline σε συνθετικά σύνολα δεδομένων [4].

2.3 Παρόμοια προβλήματα

2.3.1 Convex Hull

Το πρόβλημα του Ελάχιστου Κυρτού Πολυγώνου (Convex Hull) ενός συνόλου *P* με *n* σημεία, αφορά την εύρεση του πολυγώνου με κορυφές σημεία του συνόλου *P* και το οποίο περιλαμβάνει όλα τα σημεία του *P*. Η διαφορά του με το skyline είναι ότι ορίζει μια περιοχή ενδιαφέροντος και όχι διακριτά σημεία.

Είναι αντιληπτό ότι αν *S* είναι το σύνολο των σημείων που προκύπτει από την ένωση όλων των skylines μεγιστοποίησης και ελαχιστοποίησης του ίδιου προβλήματος και *C* το σύνολο των σημείων του Convex Hull, τότε $C \subseteq S$ όπως φαίνεται και στο Σχήμα 2.5.



Σχήμα 2.6: Ελάχιστο Κυρτό Πολύγωνο (ενωμένα σημεία) σε σχέση με όλα τα skylines (μαυρισμένα σημεία).

2.3.2 Maximal Vectors

Το πρόβλημα των Μέγιστων Διανυσμάτων (Maximal Vectors) [5] απασχολεί την υπολογιστική γεωμετρία και αφορά την εύρεση του υποσυνόλου των διανυσμάτων που δεν κυριαρχούνται από τα υπόλοιπα διανύσματα ενός συνόλου. Η έννοια της κυριαρχίας είναι ακριβώς η ίδια με το skyline, δηλαδή ένα διάνυσμα κυριαρχεί έναντι ενός άλλου όταν όλες οι τιμές των επιμέρους στοιχείων του είναι τουλάχιστον ίσες και μία τουλάχιστον τιμή είναι μεγαλύτερη.

Οι αλγόριθμοι που έχουν προταθεί για την εύρεση των μέγιστων διανυσμάτων έχουν άμεση εφαρμογή και στην εύρεση του skyline [6].

2.3.3 Pareto Front

Εκτός από το πρόβλημα των μέγιστων διανυσμάτων, ο υπολογισμός του skyline ταυτίζεται επίσης με το μέτωπο ή σύνορο Pareto (Pareto Front ή Pareto Frontier), έννοια η οποία προήλθε από τον Ιταλό οικονομολόγο Vilfredo Pareto [7]. Χρησιμοποιείται ευρέως σε τομείς όπως η μικροοικονομία και η επιχειρησιακή έρευνα για την περιγραφή κοινωνικοοικονομικών όρων όπως η ευημερία.

Δεδομένου ενός συνόλου που τα επιμέρους στοιχεία του αποτελούνται από μια σειρά πόρων με διαφορετική κατανομή, μια κατανομή ονομάζεται βέλτιστη (ή αποτελεσματική) κατά Pareto όταν δεν υπάρχει άλλη εφικτή κατανομή τέτοια ώστε να βελτιώνει τη θέση ενός στοιχείου χωρίς να χειροτερεύει τη θέση ενός άλλου. Μια κατανομή ονομάζεται κυρίαρχη έναντι μιας άλλης κατανομής αν είναι τόσο καλή σε όλα τα κριτήρια και αυστηρά καλύτερη σε ένα τουλάχιστον κριτήριο από την άλλη κατανομή Η βέλτιστη κατανομή είναι αυτή που δεν κυριαρχείται από καμία άλλη κατανομή. Το σύνολο των βέλτιστων (μη κυριαρχούμενων) κατανομών είναι το μέτωπο Pareto.

2.4 Εφαρμογές

Ο τελεστής skyline βρίσκει εφαρμογή οπουδήποτε απαιτείται η λήψη αποφάσεων με βάση πολλαπλά κριτήρια.

Μια τυπική περίπτωση εφαρμογής πολυκριτηριακής απόφασης είναι τα συστήματα συστάσεων. Ένα τέτοιο σύστημα μελέτησαν οι Kishida et al. [8] οι οποίοι επιπλέον πρότειναν τη χρήση μιας συνάρτησης βαθμολόγησης με βάση την ανατροφοδότηση από τους χρήστες ή την πυκνότητα της κατανομής των στοιχείων στην κάθε διάσταση.

Οι Bousnina et al. [9] πρότειναν τη χρήση ενός τελεστή skyline για τη δημοφιλή πλατφόρμα Tripadvisor, λαμβάνοντας υπόψη τις κριτικές των χρηστών παράλληλα με την αξιοπιστία του κάθε χρήστη.

Οι Che et al. [10] μελέτησαν την εφαρμογή του τελεστή skyline σε ιατρικά δεδομένα όπως το κόστος εξέτασης, το κόστος φαρμακευτικής αγωγής και το κόστος εισαγωγής, για την επιλογή της καταλληλότερης θεραπείας με στόχο την αποφυγή του λεγόμενου overtreatment, δηλαδή της άσκοπης θεραπείας πέραν της αναγκαίας.

Οι Mutlu et al. [11] χρησιμοποίησαν έναν τελεστή skyline για την εκτίμηση της επιρρέπειας ενός εδάφους στις κατολισθήσεις. Συγκεκριμένα συμπεριέλαβαν χαρακτηριστικά του εδάφους όπως ο προσανατολισμός, η κλίση, το υψόμετρο, το βάθος, η διάβρωση και η απόσταση από την απορροή.

Οι Alem et al. [12] παρουσίασαν ένα σύστημα ανίχνευσης εισβολής στο οποίο χρησιμοποιείται ο τελεστής skyline για την επιλογή του καλύτερου ταξινομητή από ένα πλήθος τεχνικών εξόρυξης δεδομένων με στόχο τη μεγιστοποίηση της ακρίβειας και του βαθμού ανίχνευσης παράλληλα με την ελαχιστοποίηση του βαθμού εσφαλμένου συναγερμού.

Οι Alguliyev et al. [13] πρότειναν μια λύση για την ανάθεση εκτέλεσης εργασιών από κινητές συσκευές στο cloud με την επιλογή της βέλτιστης εικονικής μηχανής να γίνεται μέσω του τελεστή skyline λαμβάνοντας υπόψη το κόστος και το χρόνο εκτέλεσης.

Οι Kertiou et al. [14] χρησιμοποίησαν δυναμικά ερωτήματα skyline για την επιλογή των αισθητήρων σε μια αρχιτεκτονική IoT που ταιριάζουν καλύτερα με μια σειρά από καθορισμένα κριτήρια όπως η διαθεσιμότητα, ακρίβεια, αξιοπιστία, χρόνος απόκρισης και κόστος.

Οι Li et al. [15] σχεδίασαν ένα σύστημα για τη σύνθεση υπηρεσιών διαδικτύου μέσω της επιλογής τους από έναν τελεστή skyline, ενώ οι Ouadah et al. [16] πρότειναν τη χρήση ενός τελεστή skyline για την κατάταξη των υπηρεσιών διαδικτύου με βάση το χρόνο απόκρισης, τη διαθεσιμότητα, το κόστος και την αξιοπιστία.

Οι Wang et al. [17] πρότειναν ένα σύστημα ανίχνευσης βλάβης πραγματικού χρόνου βασισμένο στον τελεστή skyline για μηχανήματα της βιομηχανίας χημικών.

Οι Asri και Anissa [18] χρησιμοποίησαν τον τελεστή skyline για την επιλογή της βέλτιστης οδικής διαδρομής με βάση τις προτιμήσεις του χρήστη σε χαρακτηριστικά όπως ο τύπος και η ποιότητα του οδοστρώματος, η απόσταση και το μποτιλιάρισμα.

Οι Wan et al. [19] πρότειναν τη χρήση του τελεστή skyline για την ανάλυση προϊόντων και τη δημιουργία άλλων ανταγωνιστικών ως προς αυτά.

Οι Zhou et al. [20], έχοντας ένα σύνολο δεδομένων εκπτωτικών προϊόντων, μελέτησαν την εύρεση των βέλτιστων συνδυασμών προϊόντων του skyline που ικανοποιούν τα κριτήρια ενός αγοραστή όπως τα συνολικά χρήματα και η μέγιστη έκπτωση.

Οι Sidiropoulos et al. [21] εφάρμοσαν τον τελεστή skyline σε μια σειρά από δείκτες αξιολόγησης ερευνητικών έργων με στόχο την εξαγωγή των καλύτερων ερευνητών βάση του συνόλου αυτών των δεικτών.

Οι Stoupas et al. [22] εφάρμοσαν επαναληπτικά τον τελεστή skyline με στόχο την κατάταξη πανεπιστημιακών ιδρυμάτων με βάση τις διδακτικές και ερευνητικές τους επιδόσεις.

Οι Zheng και Zhang [23] χρησιμοποίησαν επίσης επαναληπτικά τον τελεστή skyline για την πρόβλεψη μελλοντικών στοιχείων του skyline σε μεταβαλλόμενα δεδομένα. Συγκεκριμένα, στόχος αυτής της μεθόδου ήταν η ανακάλυψη των μελλοντικών πρωταγωνιστών στα κοινωνικά δίκτυα.

Ο τελεστής skyline έχει χρησιμοποιηθεί επίσης για το φιλτράρισμα αβέβαιων δεδομένων σε μορφή RDF [24], για τη βελτιστοποίηση της συσταδοποίησης γράφων κατά την εξόρυξη δεδομένων [25] αλλά και για την επίλυση του προβλήματος της δρομολόγησης οχημάτων (Vehicle Routing Problem) [26].

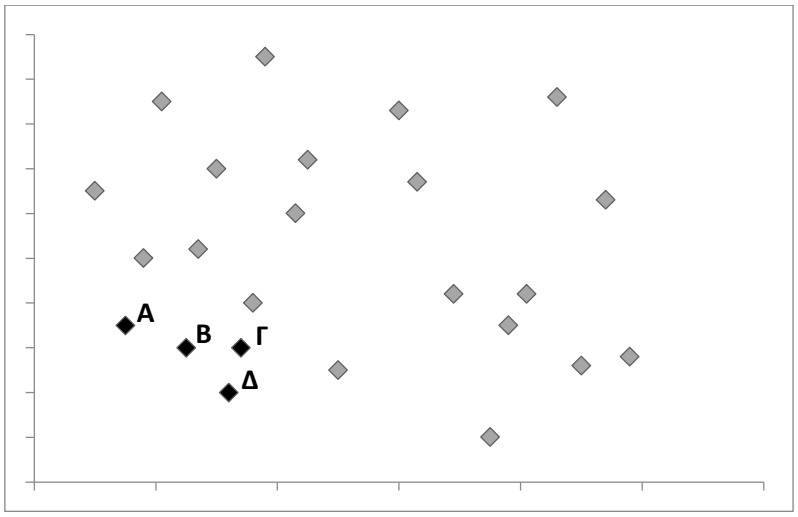
2.5 Επεκτάσεις

2.5.1 Top-k Κυρίαρχα Ερωτήματα

Δεδομένου ενός συνόλου δεδομένων D με τιμές στον d-διάστατο χώρο R^d και μιας μονότονης συνάρτησης $F: R^d \rightarrow R$, ένα ερώτημα top-k επιστρέφει τα k στοιχεία με τη μικρότερη τιμή της F. Αν ως συνάρτηση χρησιμοποιηθεί η $\mu(p) = |\{ p' \in D \mid p \succ p' \}|$, τότε το ερώτημα top-k θα επιστρέψει τα k στοιχεία που κυριαρχούν έναντι των περισσότερων άλλων στοιχείων στο σύνολο δεδομένων [27]. Το συγκεκριμένο ερώτημα είναι το λεγόμενο top-k κυρίαρχο ερώτημα (top-k dominating query).

Με βάση το παράδειγμα που χρησιμοποιήθηκε στις προηγούμενες υποενότητες, ένα top-4 κυρίαρχο ερώτημα θα επιστρέψει τα στοιχεία που φαίνονται στο Σχήμα 2.6. Συγκεκριμένα, τα στοιχεία Α και Δ κυριαρχούν έναντι 15 άλλων στοιχείων, το Β έναντι 14 άλλων στοιχείων και το Γ έναντι 11 άλλων στοιχείων. Ενδιαφέρον παρουσιάζει το γεγονός ότι ένα top-k κυρίαρχο ερώτημα μπορεί να επιστρέψει μεταξύ άλλων και στοιχεία τα οποία δεν ανήκουν στο skyline. Στο παράδειγμα του σχήματος 2.6 ένα τέτοιο στοιχείο είναι το στοιχείο Γ.

Τα top-k κυρίαρχα ερωτήματα συνδυάζουν τα πλεονεκτήματα των top-k ερωτημάτων και των ερωτημάτων skyline, χωρίς παράλληλα να παρουσιάζουν τα μειονεκτήματά τους. Αυτό το επιτυγχάνουν περιορίζοντας το μέγεθος των ενδιαφερόντων στοιχείων, καταργώντας παράλληλα την ανάγκη για επιλογή της συνάρτησης βαθμολόγησης από το χρήστη.



Σχήμα 2.7: Αποτέλεσμα top-4 κυρίαρχου ερωτήματος.

2.5.2 Δυναμικό Skyline

Το δυναμικό skyline αποτελείται από τα στοιχεία ενός συνόλου δεδομένων που δεν κυριαρχούνται από κανένα άλλο σε σχέση με ένα δεδομένο στοιχείο.

Ειδικότερα, δεδομένου ενός *d*-διάστατου συνόλου δεδομένων *S* και ενός στοιχείου *s*, το δυναμικό skyline ως προς το *s* αποτελείται από όλα τα στοιχεία του *S* που δεν είναι δυναμικά κυριαρχούμενα από κανένα άλλο στοιχείο [28].

Ένα στοιχείο p κυριαρχεί δυναμικά έναντι ενός στοιχείου q σε σχέση με το s και συμβολίζεται ως $p \prec_s q$, αν και μόνο αν $|p_i - s_i| \leq |q_i - s_i|$ για κάθε i όπου $1 \leq i \leq d$ και υπάρχει i όπου $1 \leq i \leq d$ τέτοιο ώστε $|p_i - s_i| < |q_i - s_i|$.

2.5.3 Κ-κυρίαρχο Skyline

Η έννοια του κ-κυρίαρχου skyline (k-dominant skyline) παρουσιάστηκε από τους Chan et al. [29] με στόχο τον περιορισμό του μεγέθους του skyline και την παροχή ουσιαστικότερης πληροφορίας στο χρήστη.

Σύμφωνα με τον ορισμό της, ένα στοιχείο p κ-κυριαρχεί έναντι ενός άλλου στοιχείου q αν υπάρχουν $k \leq d$ διαστάσεις στις οποίες το p κυριαρχεί έναντι του q. Το κ-κυρίαρχο skyline αποτελείται από όλα τα στοιχεία που δεν είναι κ-κυριαρχούμενα από κανένα άλλο στοιχείο.

2.5.4 Skycube

Η έννοια του Skycube προτάθηκε από τους Yuan et al. [30] στην προσπάθειά τους να περιγράψουν το αποτέλεσμα που προκύπτει από την ένωση των skylines όλων των μη κενών υποσυνόλων ενός d-διάστατου συνόλου. Συγκεκριμένα, σε ένα d-διάστατο σύνολο *S* υπάρχουν 2^d-1 υποσύνολα των διαστάσεων του *S* και αντίστοιχα skylines (Πίνακας 2.1 και Πίνακας 2.2). Η ένωση όλων αυτών των skylines αποτελεί το Skycube. Με άλλα λόγια, το Skycube δίνει την πληροφορία για τα επιμέρους skylines στα οποία ανήκει κάθε πλειάδα. Για τον υπολογισμό του έχουν προταθεί πλήθος αλγορίθμων [30], [31].

Η συμπληρωματική έννοια του παραπάνω είναι το Negative Skycube [32] το οποίο δίνει την πληροφορία για τα επιμέρους skylines στα οποία δεν ανήκει η κάθε πλειάδα. Και για το συγκεκριμένο πρόβλημα έχουν προταθεί πλήθος μεθόδων υπολογισμού [33].

Πίνακας 2.1: Σύνολο δεδομένων S. [30]

Διάσταση / ***Στοιχείο***	**A**	**B**	**C**	**D**
p1	4	3	2	2
p2	5	1	1	2
p3	1	4	4	1
p4	3	5	5	1
p5	2	2	3	1

Πίνακας 2.2: Επιμέρους skylines για όλα τα υποσύνολα των διαστάσεων του S. [30]

Διαστάσεις	***Skyline***
ABCD	{p1, p2, p3, p5}
ABC	{p1, p2, p3, p5}
ABD	{p2, p3, p5}
ACD	{p1, p2, p3, p5}
BCD	{p2, p5}
AB	{p2, p3, p5}
AC	{p1, p2, p3, p5}
AD	{p3}

Διαστάσεις	***Skyline***
BC	{p2}
BD	{p2, p5}
CD	{p2, p5}
A	{p3}
B	{p2}
C	{p2}
D	{p3, p4, p5}

2.5.5 Πιθανοτικό Skyline

Η ιδέα του πιθανοτικού skyline (Probabilistic Skyline – P-Skyline) μαζί με μια σειρά από αλγορίθμους για τον υπολογισμό της, προτάθηκε από τους Pei et al. [34] με στόχο να δώσει λύση στο πρόβλημα του υπολογισμού του skyline σε αβέβαια δεδομένα.

Σε ένα σύνολο από αβέβαια δεδομένα, κάθε στοιχείο συνδέεται με μια συνάρτηση πυκνότητας πιθανότητας (Probability Density Function). Ουσιαστικά, υπολογίζεται η πιθανότητα ένα στοιχείο να κυριαρχεί έναντι ενός άλλου. Επομένως, η πιθανότητα να ανήκει ένα στοιχείο στο skyline είναι η πιθανότητα αυτό το στοιχείο να μην κυριαρχείται από κανένα άλλο στοιχείο. Με βάση ένα κατώφλι πιθανότητας p $(0 \leq p \leq 1)$, το πιθανοτικό skyline είναι το σύνολο των αβέβαιων στοιχείων καθένα εκ των οποίων λαμβάνει μια πιθανότητα, τουλάχιστον p, να ανήκει στο skyline.

2.5.6 Αντίστροφο Skyline

Το αντίστροφο skyline (Reverse Skyline), που προτάθηκε από τους Dellis και Seeger [35], αποτελείται από τα στοιχεία που το δυναμικό τους skyline περιέχει ένα δεδομένο στοιχείο ενδιαφέροντος *p*.

Στόχος του αντίστροφου skyline είναι ο εντοπισμός της επιρροής ενός δεδομένου στοιχείου ενδιαφέροντος σε ένα πολυδιάστατο σύνολο δεδομένων σε σχέση με ένα διάνυσμα αποστάσεων. Ως επιρροή νοείται η κατάσταση όπου δεδομένου ενός στοιχείου *p*, υπάρχουν κάποια στοιχεία για τα οποία το *p* ανήκει στο δυναμικό skyline τους.

Πρακτικά, το αντίστροφο skyline αποδεικνύεται χρήσιμο σε περιπτώσεις μάρκετινγκ και προώθησης όπου το ζητούμενο είναι ο εντοπισμός των πιθανά ενδιαφερόντων στοιχείων με βάση ένα δεδομένα ενδιαφέρον στοιχείο.

2.5.7 Ομαδικό Skyline

Ένα ομαδικό skyline (Group Skyline – G-Skyline) αναπαριστά τις ομάδες στοιχείων ενός συνόλου δεδομένων που δεν κυριαρχούνται από άλλες ομάδες [36].

Συγκεκριμένα, σε ένα *d*-διάστατο σύνολο δεδομένων *P* αποτελούμενο από *n* στοιχεία, μια ομάδα *k* στοιχείων $G = \{p_1, p_2, \ldots, p_k\}$ κυριαρχεί έναντι μιας άλλης ομάδας *k* στοιχείων $G' = \{p'_1, p'_2, \ldots, p'_k\}$ αν υπάρχουν δύο μεταθέσεις των *k* στοιχείων των *G* και *G'*, $G = \{p_{u_1}, p_{u_2}, \ldots, p_{u_k}\}$ και $G' = \{p'_{v_1}, p'_{v_2}, \ldots, p'_{v_k}\}$ τέτοιες ώστε $p_{u_i} \preccurlyeq p'_{v_i}$ για κάθε i όπου $(1 \leq i \leq k)$ και $p_{u_i} \prec p'_{v_i}$ για τουλάχιστον ένα *i*. Το ομαδικό skyline *k* στοιχείων αποτελείται από όλες τις ομάδες *k* στοιχείων που δεν κυριαρχούνται από καμία άλλη ομάδα ίδιου μεγέθους.

Όπως και για το απλό skyline, έτσι και για το ομαδικό ισχύει η ασυμμετρία και η μεταβατική ιδιότητα. Μια περιεκτική σύγκριση και αξιολόγηση των αλγορίθμων υπολογισμού ομαδικού skyline έχει γίνει από τους Zhu et al. [37]

Το ομαδικό skyline λύνει το πρόβλημα της εύρεσης κυρίαρχων ομάδων στοιχείων και όχι απλά μεμονωμένων στοιχείων. Αυτό είναι χρήσιμο σε περιπτώσεις όπου αναζητείται, για παράδειγμα από ένα σύνολο δεδομένων με στατιστικά ποδοσφαιριστών, η καλύτερη ομάδα ποδοσφαιριστών και όχι ο καλύτερος ποδοσφαιριστής μεμονωμένα.

2.5.8 Χωρικό Skyline

Η έννοια του χωρικού skyline και των αντίστοιχων ερωτημάτων (Spatial Skyline Query - SSQ) παρουσιάστηκε από τους Sharifzadeh και Shahabi [38] και αφορά τα στοιχεία ενός συνόλου δεδομένων που δεν κυριαρχούνται από κανένα άλλο στοιχείο σε σχέση με τα χωρικά τους χαρακτηριστικά.

Συγκεκριμένα, κάθε χωρικό χαρακτηριστικό ενός στοιχείου *p* αντιπροσωπεύει μια απόσταση από ένα στοιχείο ενδιαφέροντος *q* με βάση μια συνάρτηση απόστασης $D(p, q)$, για παράδειγμα την Ευκλείδεια απόσταση. Δεδομένου ενός συνόλου *d*-διάστατων στοιχείων ενδιαφέροντος $Q = \{q_1, q_2, \ldots, q_n\}$ και δύο στοιχείων *p* και *p'* στον *d*-διάστατο χώρο, το *p* κυριαρχεί χωρικά έναντι του *p'* σε σχέση με το Q, αν και μόνο αν $D(p, q_i) \leq D(p', q_i)$ για κάθε $q_i \in Q$ και $D(p, q_j) < D(p', q_j)$ για κάποιο $q_j \in Q$. Πρακτικά, ένα στοιχείο *p* κυριαρχεί χωρικά έναντι ενός στοιχείου *p'* αν και μόνο αν κάθε στοιχείο ενδιαφέροντος q_i είναι πιο κοντά στο *p* από ότι στο *p'*.

Η χωρική κυριαρχία βρίσκει εφαρμογή στα προβλήματα όπου η βέλτιστη λύση σχετίζεται με την απόσταση από συγκεκριμένα στοιχεία. Τέτοια προβλήματα είναι για παράδειγμα ο σχεδιασμός εγκαταστάσεων, η διαχείριση κρίσεων και ο σχεδιασμός ταξιδιών και εκδηλώσεων.

Κεφάλαιο 3ο: Αλγόριθμοι για την εύρεση skyline

3.1 Εισαγωγή

Από τη στιγμή που παρουσιάστηκε η έννοια του τελεστή skyline μέχρι και σήμερα, πλήθος αλγορίθμων έχουν προταθεί με σκοπό τη βελτιστοποίησή του. Κάποιοι από αυτούς χρησιμοποιούν ευρετήρια ενώ άλλοι όχι. Η χρησιμοποίηση ευρετηρίων συμβάλλει στην καλύτερη απόδοση των αλγορίθμων αφού αποφεύγεται η προσπέλαση του συνόλου των δεδομένων, απαιτεί όμως προηγούμενη ευρετηρίαση των δεδομένων και έχει περιορισμούς.

Ένας άλλος διαχωρισμός που μπορεί να γίνει είναι ανάμεσα στους αλγορίθμους που χρησιμοποιούν ταξινόμηση, με στόχο το γρήγορο αποκλεισμό των κυριαρχούμενων στοιχείων, και τους αλγορίθμους που επιμερίζουν το σύνολο δεδομένων με στόχο το γρήγορο υπολογισμό του skyline σε μικρότερα τμήματά του. Η ταξινόμηση του συνόλου δεδομένων μειώνει γενικά τον αριθμό των απαιτούμενων συγκρίσεων για τον καθορισμό του skyline, η αποτελεσματικότητά της, όμως, εξαρτάται άμεσα από την επιλογή της συνάρτησης ταξινόμησης και το εκάστοτε κατώφλι. Οι αλγόριθμοι που επιμερίζουν το σύνολο δεδομένων χάνουν σε απόδοση καθώς αυξάνονται οι διαστάσεις.

Η βασική μετρική για την επίδοση ενός αλγορίθμου εύρεσης skyline είναι ο αριθμός των συγκρίσεων που επιτελεί. Όσο λιγότερες συγκρίσεις πραγματοποιούνται τόσο μικρότερος είναι ο υπολογιστικός φόρτος αλλά και η απαιτούμενη μνήμη. Η απλοϊκή προσέγγιση απαιτεί $n*(n-1)$ συγκρίσεις, έχει δηλαδή τετραγωνική πολυπλοκότητα. Ο αριθμός των απαιτούμενων συγκρίσεων μπορεί να μειωθεί χρησιμοποιώντας ευρετηρίαση, ταξινόμηση και τη μεταβατική ιδιότητα της κυριαρχίας.

3.2 Block Nested Loops

Ο αλγόριθμος Block Nested Loops (BNL) αποτελεί την απλούστερη προσέγγιση στο πρόβλημα όπως παρουσιάστηκε από τους Borzsony et al. [1]. Αφού δεσμευτεί αρχικά ένας χώρος (παράθυρο) στην κύρια μνήμη, ο αλγόριθμος συγκρίνει σειριακά όλες τις πλειάδες του συνόλου των δεδομένων μεταξύ τους. Αυτό επιτυγχάνεται με την αρχική τοποθέτηση της πρώτης πλειάδας στο παράθυρο και τη μετέπειτα σύγκριση όλων των υπόλοιπων πλειάδων με αυτή. Αν η πλειάδα που συγκρίνεται κυριαρχεί έναντι μιας ή περισσότερων πλειάδων στο παράθυρο, τότε οι πλειάδες αυτές αφαιρούνται από το παράθυρο και πρακτικά αποκλείονται από τις επόμενες επαναλήψεις του αλγορίθμου. Αντίθετα, αν η πλειάδα που συγκρίνεται κυριαρχείται από μία τουλάχιστον πλειάδα στο παράθυρο, τότε δεν έχει νόημα η σύγκρισή της με τις υπόλοιπες πλειάδες του παραθύρου αφού είναι βέβαιο ότι δεν ανήκει στο skyline. Τέλος, αν η πλειάδα που συγκρίνεται είναι μη συγκρίσιμη με όλες τις πλειάδες στο παράθυρο, τότε προστίθεται κι αυτή στο παράθυρο. Στην περίπτωση όπου το παράθυρο στη μνήμη γεμίσει, οι υπόλοιπες κυρίαρχες ή μη συγκρίσιμες πλειάδες τοποθετούνται σε ένα προσωρινό αρχείο και χρησιμοποιούνται ως είσοδος στις επόμενες επαναλήψεις του αλγορίθμου.

Ο αλγόριθμος BNL είναι ικανοποιητικός αν το μέγεθος του skyline είναι αρκετά μικρό ώστε να μην προκαλέσει κορεσμό του παραθύρου και να ολοκληρωθεί μετά από μία επανάληψη. Η απόδοσή του, όμως, εξαρτάται άμεσα από το πόσο πολυδιάστατα είναι τα δεδομένα καθώς και από τη διάταξή τους. Συγκεκριμένα, η απόδοση μειώνεται κατακόρυφα όσο περισσότερες είναι οι διαστάσεις και όσο πιο αρνητική συσχέτιση υπάρχει στη διάταξη των δεδομένων.

Στη χειρότερη περίπτωση, η πολυπλοκότητα του αλγορίθμου BNL είναι $O(n^2)$ όπου n είναι το μέγεθος του συνόλου δεδομένων.

Αλγόριθμος BNL

Είσοδος: Ένα σύνολο δεδομένων *D*

Έξοδος: Το skyline του *D*

1. Αρχικοποίησε ένα κενό σύνολο $S = \emptyset$
2. Μετακίνησε το πρώτο στοιχείο του συνόλου δεδομένων στο *S*
3. Για κάθε $p \in D$
4. **Για κάθε** $q \in S$
5. **Αν** $p \succ q$ **Τότε**
6. Διέγραψε το *q* από το *S*
7. **Αν** $q \succ p$ **Τότε**
8. Σταμάτα τις συγκρίσεις του *p*
9. **Αν** ούτε $p \succ q$ για κάθε *q*, ούτε $q \succ p$ για κάθε *q* **Τότε**
10. Πρόσθεσε το *p* στο *S*
11. Επέστρεψε το *S*

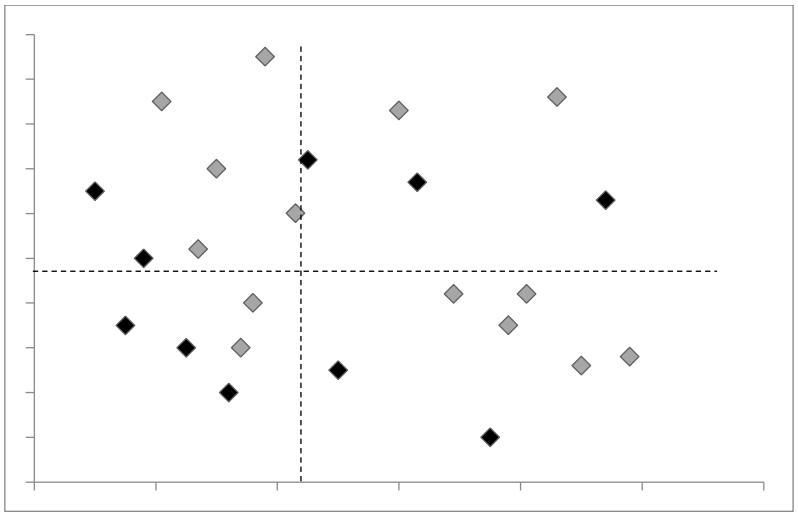
3.3 Divide and Conquer

Μαζί με την απλοϊκή προσέγγιση του BNL, οι Borzsony et al. [1] παρουσίασαν έναν ακόμα αλγόριθμο του τύπου «διαίρει και βασίλευε» (Divide and Conquer - DC). Ο αλγόριθμος DC διαιρεί αναδρομικά το σύνολο δεδομένων σε μικρότερα τμήματα ώστε κάθε ένα από αυτά να χωράει στην κύρια μνήμη. Στη συνέχεια, υπολογίζεται το skyline για τα επιμέρους τμήματα και τελικά το skyline για ολόκληρο το σύνολο δεδομένων προκύπτει από τη συγχώνευση των skylines των επιμέρους τμημάτων.

Όπως ο BNL έτσι και ο DC δεν είναι ιδιαίτερα αποδοτικός αφού απαιτεί αρκετά μεγάλη κύρια μνήμη και πρέπει πρώτα να περάσει από όλα τα δεδομένα για να καταλήξει σε τουλάχιστον ένα δεδομένο που να ανήκει στο skyline. Επιπλέον, όσο αυξάνεται το μέγεθος του συνόλου δεδομένων και κυρίως οι διαστάσεις, η απόδοσή του μειώνεται αφού ένα στοιχείο που δεν ανήκει στο skyline είναι πιθανότερο να ανήκει στο τοπικό skyline.

Για την αντιμετώπιση του προβλήματος της περιορισμένης μνήμης, στην ίδια έρευνα προτάθηκαν και δύο βελτιστοποιήσεις του αλγορίθμου ονόματι *M-Way Partitioning* και *Early Skyline*.

Στη χειρότερη περίπτωση, η πολυπλοκότητα του αλγορίθμου DC είναι $O(n*(logn)^{d-2}) + O(n*logn)$ όπου *n* είναι το μέγεθος του συνόλου δεδομένων και *d* ο αριθμός των διαστάσεων.



Σχήμα 3.1: Διαίρεση του συνόλου δεδομένων από τον αλγόριθμο DC.

***Αλγόριθμος* Divide and Conquer**

Είσοδος: Ένα σύνολο δεδομένων *D*

Έξοδος: Το skyline του *D*

1. Διαίρεσε το *D* χρησιμοποιώντας τη διάμεση τιμή μιας διάστασης *Di*. Έστω D_{Hi} το υποσύνολο του *D* με τιμή στη διάσταση *Di* μεγαλύτερη της διάμεσης και D_{Li} το υποσύνολο των υπόλοιπων στοιχείων.
2. Υπολόγισε το skyline των υποσυνόλων D_{Hi} και D_{Li} εφαρμόζοντας αναδρομικά το Βήμα 1.
3. Συγχώνευσε τα skylines των D_{Hi} και D_{Li}

3.4 Bitmap

Ένα από τα μειονεκτήματα των BNL και DC που παρουσιάστηκαν στις προηγούμενες υποενότητες είναι το πέρασμα του αλγορίθμου από το σύνολο των δεδομένων πριν την επιστροφή τουλάχιστον ενός στοιχείου του skyline. Για να το αντιμετωπίσουν αυτό, οι Tan et al. [39] πρότειναν ένα αλγόριθμο που βασίζεται σε ευρετήρια. Ο αλγόριθμος αυτός δέχεται τα δεδομένα ως ακολουθίες από bit και εκμεταλλεύεται την ταχύτητα της δυαδικής πράξης AND. Δεν χρειάζεται να περάσει από το σύνολο των δεδομένων για να επιστρέψει κάποια αποτελέσματα, δηλαδή υπολογίζει το skyline προοδευτικά καθώς διαβάζει τα δεδομένα.

Κάθε ένα από τα δεδομένα αντιστοιχίζεται σε ένα διάνυσμα από *m* bits όπου *m* είναι το άθροισμα του αριθμού των διακριτών τιμών καθεμιάς από τις *d* διαστάσεις, δηλαδή $m = \sum_{i=1}^{d} ki$. Στην περίπτωση

ενός προβλήματος ελαχιστοποίησης, αν η διάσταση *i* παριστάνεται από k_i bits τότε για την κατά σειρά j_i μικρότερη τιμή τα $k_i - j_i + 1$ πιο σημαντικά bits παίρνουν την τιμή 1 και τα υπόλοιπα την τιμή 0. Για παράδειγμα, αν μια διάσταση αποτελείται από 10 διακριτές τιμές όπου η τιμή 30 είναι η τρίτη μικρότερη σε αυτή τη διάσταση, τότε η δυαδική αναπαράσταση αυτής της τιμής θα αποτελείται από 10 bits εκ των οποίων τα 10 - 3 + 1 = 8 πιο σημαντικά bits θα είναι άσσοι και τα υπόλοιπα μηδενικά, δηλαδή 1111111100 (Πίνακας 3.1). Για την κατάταξη ή μη ενός στοιχείου από τα δεδομένα στο skyline, εφαρμόζεται αρχικά η δυαδική πράξη AND σε *d* bit-slices V_x, V_y … V_d τα οποία σχηματίζονται μετά την αντιπαραβολή των αντίστοιχων λιγότερο σημαντικών bits από την κωδικοποίηση κάθε στοιχείου. Στη συνέχεια, εφαρμόζεται η δυαδική πράξη OR σε *d* bit bit-slices τα οποία σχηματίζονται μετά την αντιπαραβολή των κατά μία θέση λιγότερο σημαντικών bits σε σχέση με την πράξη AND που προηγήθηκε. Για παράδειγμα, αν η τιμή ενός στοιχείου στη διάσταση *i* είναι η τρίτη μικρότερη, τότε το V_i κατά την πράξη AND θα αποτελείται από τα τρίτα λιγότερο σημαντικά bits των τιμών αυτής της διάστασης, ενώ κατά την πράξη OR θα αποτελείται από τα δεύτερα λιγότερο σημαντικά bits. Αντίστοιχα και για τις υπόλοιπες διαστάσεις. Τελικά, εφαρμόζεται και πάλι η δυαδική πράξη AND, αυτή τη φορά μεταξύ των δύο δυαδικών αριθμών που παρήχθησαν από τα δύο προηγούμενα βήματα. Αν το αποτέλεσμα αυτής της πράξης αποτελείται μόνο από μηδενικά, τότε το στοιχείο αυτό ανήκει στο skyline, ειδάλλως όχι.

Ακόμα κι αν ο αλγόριθμος Bitmap υπολογίζει προοδευτικά το skyline, πάλι είναι αναγκαίο να περάσει από το σύνολο των δεδομένων για να υπολογίσει ολόκληρο το skyline, πράγμα που δεν τον καθιστά ιδιαίτερα αποδοτικό. Επιπλέον, η σειρά με την οποία επιστρέφονται τα στοιχεία του skyline δεν μπορεί να καθοριστεί από το χρήστη.

Πίνακας 3.1: Δυαδική αναπαράσταση δεδομένων και bit-slices.

A - (50, 20)	(1111110**0**00, 111**1**1110)
B - (30, 50)	(1111111**1**00, 100**0**0000)
C - (10, 80)	(1111111**1**11, 111**1**0000)
D - (60, 80)	(1111100**0**00, 111**1**0000)
E - (100, 20)	(1000000**0**00, 111**1**1110)
F - (20, 30)	(1111111**1**10, 111**1**1100)
G - (40, 70)	(1111111**0**00, 110**0**0000)
H - (80, 40)	(1110000**0**00, 111**1**1000)
I - (90, 10)	(1100000**0**00, 111**1**1111)
J - (70, 60)	(1111000**0**00, 111**0**0000)

Αλγόριθμος Bitmap [39]

Είσοδος: Ένα σύνολο δεδομένων *D*

Έξοδος: Το skyline του *D*

1. Για κάθε $p \in D$
2. Έστω p_i είναι η κατά σειρά *n* διακριτή τιμή στη διάσταση *i*
3. A ← BitSlice(n_1, 1)
4. **Για κάθε** διάσταση i > 1
5. A ← A & BitSlice(n_i, i)
6. B ← BitSlice(n_1 - 1, 1)
7. **Για κάθε** διάσταση i > 1
8. B ← B | BitSlice(n_i - 1, i)
9. C ← A & B
10. **Αν** C == 0 **Τότε**
11. **Επέστρεψε** το p

3.5 Index

Μαζί με τον αλγόριθμο Bitmap, οι Tan et al. [39] πρότειναν έναν ακόμα αλγόριθμο, τον Index. Ο αλγόριθμος αυτός χρησιμοποιεί ένα ειδικό B-δέντρο για την ευρετηρίαση όλων των στοιχείων του συνόλου δεδομένων.

Ένα στοιχείο p του συνόλου δεδομένων ανήκει στην *i*-οστή λίστα ($1 \leq i, j \leq d$) αν η τιμή του στη διάσταση *i* είναι η μικρότερη ή η μεγαλύτερη από τις τιμές όλων των υπόλοιπων διαστάσεών του, δηλαδή $p[i] \leq p[j]$ ή $p[i] \geq p[j]$ για κάθε $i \neq j$. Σε κάθε λίστα, τα στοιχεία ομαδοποιούνται και ταξινομούνται με βάση την ελάχιστη ή μέγιστη τιμή σε αυτή τη διάσταση. Στοιχεία με ίδια ελάχιστη ή μέγιστη τιμή ομαδοποιούνται μαζί. Ο δείκτης της κάθε ομάδας είναι η μικρότερη ή μεγαλύτερη τιμή του στοιχείου που αντιπροσωπεύει. Για παράδειγμα, με βάση τα ίδια στοιχεία που χρησιμοποιήθηκαν στο παράδειγμα της προηγούμενης υποενότητας, οι λίστες που δημιουργεί ο αλγόριθμος Index φαίνονται στον Πίνακα 3.2. Ο αλγόριθμος εντοπίζει τα στοιχεία που ανήκουν στο skyline εξετάζοντας τις ομάδες με αύξουσα ή φθίνουσα σειρά δείκτη, ενώ αν μια ομάδα έχει περισσότερα από ένα στοιχεία υπολογίζει το τοπικό skyline το οποίο στη συνέχεια ενσωματώνει στο καθολικό skyline.

Ουσιαστικά δημιουργούνται τόσες λίστες όσες και διαστάσεις, όπου κάθε λίστα αποτελείται από τα στοιχεία με τη μέγιστη ή ελάχιστη τιμή σε αυτή τη διάσταση, κατά φθίνουσα ή αύξουσα ταξινόμηση, αναλόγως αν πρόκειται για πρόβλημα μεγιστοποίησης ή ελαχιστοποίησης αντίστοιχα. Ο αλγόριθμος, εξετάζοντας με τη σειρά τα στοιχεία με βάση τους δείκτες, προσθέτει στοιχεία στο skyline φτάνοντας

κάποια στιγμή σε ένα σημείο όπου πλέον δεν είναι αναγκαίο να ελέγξει άλλα στοιχεία καθώς είναι σίγουρο ότι αυτά δεν ανήκουν στο skyline.

Ο αλγόριθμος Index είναι σαφώς πιο αποδοτικός από τους αλγορίθμους που παρουσιάστηκαν στις προηγούμενες υποενότητες αφού επιστρέφει γρήγορα τα σημεία του skyline, η σειρά των οποίων όμως καθορίζεται άμεσα από την ταξινόμηση των στοιχείων στο B-δέντρο κάτι το οποίο μπορεί να είναι περιοριστικό για ορισμένες εφαρμογές.

Πίνακας 3.2: Οι λίστες που δημιουργεί ο αλγόριθμος Index.

Λίστα 1		***Λίστα 2***	
C - (10, 80)	min = 10	I - (90, 10)	min = 10
F - (20, 30)	min = 20	A - (50, 20), E - (100, 20)	min = 20
B - (30, 50)	min = 30	H (80, 40)	min = 40
G - (40, 70)	min = 40	J (70, 60)	min = 60
D - (60, 80)	min = 60		

Αλγόριθμος Index [39]

Είσοδος: Ένα σύνολο δεδομένων *D*

Έξοδος: Το skyline του *D*

1. **Για κάθε** διάσταση *i*
2. $f_i \leftarrow$ True
3. $t_i \leftarrow$ traverseTreeMax(ρίζα, i)
4. $max_i \leftarrow$ maxValue(t_i)
5. $min_i \leftarrow$ minValue(t_i)
6. mn $\leftarrow max_{i=1}^{d} min_i$
7. mx $\leftarrow max_{i=1}^{d} max_i$
8. **Για κάθε** διάσταση *i*
9. **Αν** mn > max_i **Τότε**
10. $f_i \leftarrow$ False
11. j $\leftarrow$ 1
12. S $\leftarrow$ 0
13. **Για όσο** υπάρχουν ακόμα ομάδες για αναζήτηση
14. **Για κάθε** διάσταση *i*

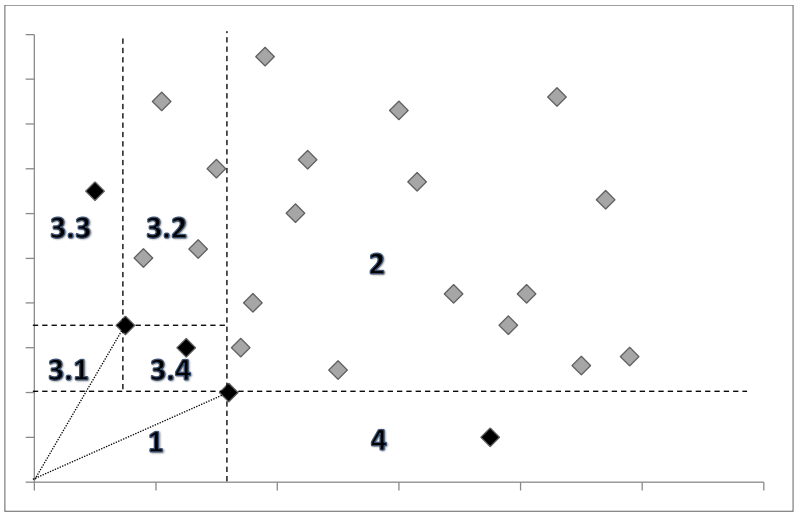
15. **Αν** max_i == mx **Τότε**
16. $P_j \leftarrow ti$
17. $S_j \leftarrow 0$
18. $t_i \leftarrow$ getNextLeftElement(t_i)
19. **Για όσο** (maxValue(t_i) == mx)
20. Mn ← max(mn, minValue(t_i))
21. $P_j \leftarrow P_j$ U t_i
22. $t_i \leftarrow$ getNextLeftElement(t_i)
23. $max_i \leftarrow$ maxValue(t_i)
24. $S_j \leftarrow$ computePartitionSkyline(P_j)
25. S ← S U computeNewSkyline(S_j, S)
26. j ← j + 1
27. mx ← $max_{i=1}^{d} max_i$
28. **Για κάθε** διάσταση i
29. **Αν** mn > max_i **Τότε**
30. $f_i \leftarrow$ False

3.6 Nearest Neighbor

Ο αλγόριθμος του Πλησιέστερου Γείτονα (Nearest Neighbor - NN) που προτάθηκε από τους Kossmann et al. [40] χρησιμοποιεί ένα R-δέντρο για να αποκλείσει μαζικά στοιχεία από το σύνολο δεδομένων, αποφεύγοντας έτσι τις περιττές συγκρίσεις.

Ο συγκεκριμένος αλγόριθμος εφαρμόζει αναδρομικά την αναζήτηση του πλησιέστερου γείτονα με βάση οποιαδήποτε μονότονη συνάρτηση απόστασης όπως η Ευκλείδεια. Αρχικά, εντοπίζει το πλησιέστερο ή πιο απομακρυσμένο στοιχείο στην αρχή των αξόνων, ανάλογα με το αν πρόκειται για πρόβλημα ελαχιστοποίησης ή μεγιστοποίησης αντίστοιχα. Το στοιχείο αυτό, το οποίο ανήκει σίγουρα στο skyline, χωρίζει το χώρο σε τέσσερις υποπεριοχές. Η πρώτη υποπεριοχή δεν περιέχει κανένα στοιχείο, η δεύτερη υποπεριοχή αποτελείται από τα στοιχεία τα οποία είναι βέβαιο πως κυριαρχούνται από αυτό το στοιχείο, ενώ οι υπόλοιπες δύο υποπεριοχές πρέπει να ερευνηθούν με αναδρομική εκτέλεση του αλγορίθμου ώστε να εξαχθεί το τελικό skyline. (Σχήμα 3.1)

Πλεονέκτημα του αλγορίθμου NN είναι ο άμεσος αποκλεισμός των στοιχείων που είναι βέβαιο πως δεν ανήκουν στο skyline. Όμως, παρουσιάζει μεγάλη επιβάρυνση για σύνολα δεδομένων με περισσότερες από δύο διαστάσεις, γεγονός που μπορεί να καταστήσει απαγορευτική την εφαρμογή του.



Σχήμα 3.2: Γραφική απεικόνιση του αλγορίθμου NN.

Αλγόριθμος Nearest Neighbor [40]

Είσοδος: Ένα σύνολο δεδομένων *D* και
μια συνάρτηση απόστασης *f*

Έξοδος: Το skyline του *D*

1. $T = \{(\infty, \infty)\}$
2. **Για όσο** ($T \neq \emptyset$)
3. (m_x, m_y) = takeElement(*T*)
4. **Αν** υπάρχει boundedNNSearch(*O*, *D*, (m_x, m_y), *f*) **Τότε**
5. (n_x, n_y) = boundedNNSearch(*O*, *D*, (m_x, m_y), *f*)
6. T = T U {(n_x, m_y), (m_x, n_y)}
7. **Επέστρεψε** το n

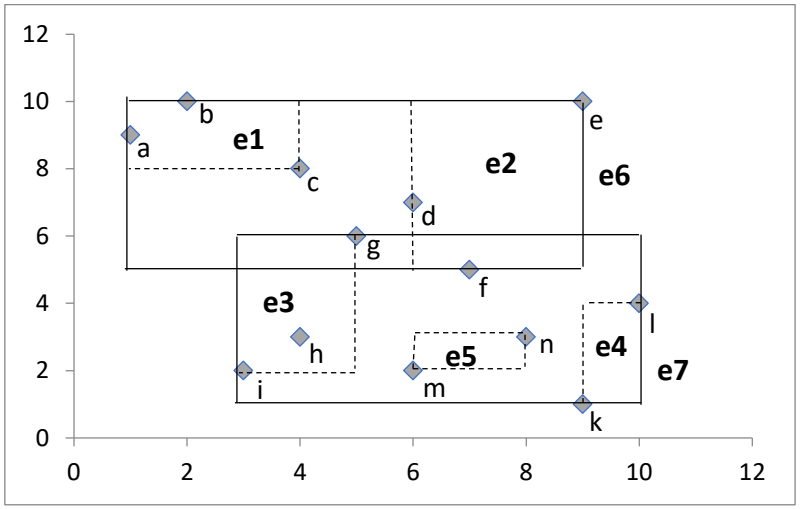
3.7 Branch and Bound Skyline

Για να ξεπεραστούν τα μειονεκτήματα του αλγορίθμου NN, οι Papadias et al. [28] πρότειναν τον αλγόριθμο Branch and Bound Skyline (BBS). Ο αλγόριθμος αυτός, όπως και ο NN που παρουσιάστηκε στην προηγούμενη υποενότητα, βασίζεται στην αναζήτηση του πλησιέστερου γείτονα χρησιμοποιώντας ένα R-δέντρο. Η διαφορά είναι ότι ο BBS διασχίζει το R-δέντρο μόνο μία φορά.

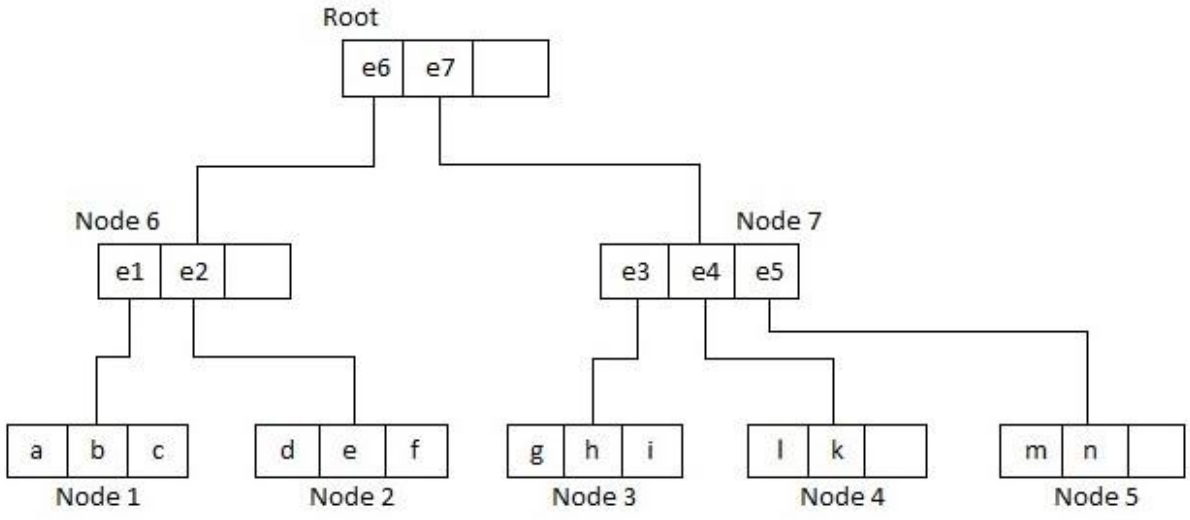
Οι κόμβοι του R-δέντρου αντιπροσωπεύουν τα αντίστοιχα Ελάχιστα Περικλείοντα Ορθογώνια (Minimum Bounding Rectangles) ενώ τα φύλλα αντιπροσωπεύουν τα στοιχεία του συνόλου δεδομένων. Ο δείκτης κάθε φύλλου είναι το άθροισμα των συντεταγμένων του και αντίστοιχα ο δείκτης για τους ενδιάμεσους κόμβους είναι το άθροισμα των συντεταγμένων της κάτω αριστερά

γωνίας του. Ο αλγόριθμος BBS διασχίζει αναδρομικά το δέντρο από τη ρίζα του, διατηρώντας ένα σωρό (heap) στη μνήμη. Ο σωρός αυτός αρχικά περιέχει τη ρίζα του δέντρου και σε κάθε επανάληψη ο κόμβος του σωρού με το μικρότερο δείκτη αντικαθίσταται από τα παιδιά του. Αν αυτά τα παιδιά αποτελούν φύλλα του δέντρου, τότε γίνεται έλεγχος κυριαρχίας μετά του οποίου τα κυρίαρχα ή μη συγκρίσιμα στοιχεία προστίθενται στο skyline ενώ τα κυριαρχούμενα διαγράφονται. Η διαδικασία συνεχίζεται μέχρις ότου αδειάσει ο σωρός. Ένα παράδειγμα της παραπάνω διαδικασίας του αλγορίθμου φαίνεται στα Σχήματα 3.3 και 3.4 αλλά και στον Πίνακα 3.3.

Παρόλο που ο BBS εγγυάται τον ελάχιστο αριθμό διασχίσεων του δέντρου για την εξαγωγή του skyline, παραμένει μη αποδοτικός για πολλές διαστάσεις.



Σχήμα 3.3: Ελάχιστα Περικλείοντα Ορθογώνια ενός συνόλου δεδομένων. [28]



Σχήμα 3.4: R-δέντρο για τα δεδομένα του Σχήματος 3.3. [28]

Πίνακας 3.3: Διάσχιση του R-δέντρου και περιεχόμενα του σωρού και του skyline σε κάθε βήμα. [28]

Ενέργεια	***Σωρός***	***Skyline***
Είσοδος στη ρίζα	(e7, 4), (e6, 6)	Ø
Επέκταση e7	(e3, 5), (e6, 6), (e5, 8), (e4, 10)	Ø
Επέκταση e3	(**i, 5**), (e6, 6), (h, 7), (e5, 8), (e4, 10), (g, 11)	{i}
Επέκταση e6	~~(h, 7), (e5, 8)~~, (e1, 9), (e4, 10), (g, 11)	{i}
Επέκταση e1	(**a, 10**), (e4, 10), (g, 11), (b, 12), (c, 12)	{i, a}
Επέκταση e4	(**k, 10**), ~~(g, 11), (b, 12), (c, 12), (l, 14)~~	{i, a, k}

Αλγόριθμος Branch and Bound Skyline [28]

Είσοδος: Ένα σύνολο δεδομένων *D*

Έξοδος: Το skyline του *D*

1. Αρχικοποίησε ένα κενό σύνολο *S* = Ø
2. Μετακίνησε τη ρίζα του *D* στο σωρό
3. **Για όσο** ο σωρός δεν είναι άδειος
4. Πάρε την κορυφαία εγγραφή *e* (κόμβος του δέντρου)
5. **Αν** η *e* κυριαρχείται από κάποιο στοιχείο του *S* **Τότε**
6. Απόρριψε το e
7. **Αλλιώς**
8. **Αν** το *e* είναι ενδιάμεση εγγραφή **Τότε**
9. **Για κάθε** παιδί e_i του *e*
10. **Αν** το e_i δεν κυριαρχείται από κανένα στοιχείο του S **Τότε**
11. Πρόσθεσε το e_i στο σωρό
12. **Αλλιώς**
13. Πρόσθεσε το e_i στο σωρό

3.8 Sort Filter Skyline

Ο αλγόριθμος Sort Filter Skyline (SFS) που προτάθηκε από τους Chomicki et al. [41] αποτελεί βελτιστοποίηση του αλγορίθμου BBS που παρουσιάστηκε στην προηγούμενη υποενότητα.

Συγκεκριμένα, ταξινομεί με αύξουσα σειρά τα δεδομένα με βάση μια μονότονη συνάρτηση. Με αυτό τον τρόπο, είναι βέβαιο πως ένα στοιχείο p που κυριαρχεί έναντι ενός σημείου q θα εξεταστεί πριν το q και επομένως ο συνολικός αριθμός των συγκρίσεων θα μειωθεί. Εκτός από αυτή τη διαφορά στην είσοδο των δεδομένων, ο αλγόριθμος SFS λειτουργεί με παρόμοιο τρόπο με τον BBS διατηρώντας και πάλι ένα σωρό στη μνήμη. Όσο αφορά τη συνάρτηση που χρησιμοποιείται στην ταξινόμηση, σύμφωνα με τους ερευνητές που πρότειναν τον αλγόριθμο, την καλύτερη απόδοση παρουσιάζει η λεγόμενη εντροπία, $E(p) = \sum_{i=1}^{d} \ln(p'[i] + 1)$, όπου $p'[i]$ είναι η κανονικοποιημένη μορφή του $p[i]$ στο διάστημα (0,1). Όσο μικρότερη είναι η τιμή της εντροπίας ενός στοιχείου, τόσο πιθανότερο είναι αυτό να πρόκειται για κυρίαρχο στοιχείο.

Στον Πίνακα 3.4, για παράδειγμα, οι τιμές του συνόλου δεδομένων κανονικοποιήθηκαν στο διάστημα (0,1) αφού διαιρέθηκαν με τον αριθμό 200. Μετά τον υπολογισμό την εντροπίας τους, και εφόσον πρόκειται για πρόβλημα ελαχιστοποίησης, τα δεδομένα θα εξεταστούν με αύξουσα σειρά εντροπίας αφού όσο μικρότερη είναι αυτή τόσο μεγαλύτερες πιθανότητες υπάρχουν να ανήκει στο skyline. Αντίστοιχα, για ένα πρόβλημα ελαχιστοποίησης τα στοιχεία θα εξετάζονταν κατά φθίνουσα σειρά εντροπίας.

Στη χειρότερη περίπτωση, η πολυπλοκότητα του αλγορίθμου SFS είναι $O(dn^2)$ όπου d είναι ο αριθμός των διαστάσεων και n είναι το μέγεθος του συνόλου δεδομένων.

Πίνακας 3.4: Κανονικοποιημένη μορφή και εντροπία ενός συνόλου δεδομένων.

Τιμές δεδομένων	***Κανονικοποιημένες τιμές***	***Εντροπία***
(50, 20)	(0.25, 0.10)	0.318
(30, 50)	(0.15, 0.25)	0.363
(10, 80)	(0.05, 0.40)	0.385
(60, 80)	(0.30, 0.40)	0.599
(100, 20)	(0.50, 0.10)	0.500
(20, 30)	(0.10, 0.15)	0.235
(40, 70)	(0.20, 0.35)	0.482
(80, 40)	(0.40, 0.20)	0.519
(90, 10)	(0.45, 0.05)	0.420
(70, 60)	(0.35, 0.30)	0.562

3.9 Linear Elimination Sort for Skyline

Ο αλγόριθμος Linear Elimination Sort for Skyline (LESS) προτάθηκε από τους Godfrey et al. [6] και αποτελεί βελτιστοποίηση του αλγορίθμου SFS. Όπως και ο αλγόριθμος SFS, έτσι και ο LESS ταξινομεί τα δεδομένα με βάση τη συνάρτηση εντροπίας, πραγματοποιώντας όμως δύο βελτιστοποιήσεις.

Η πρώτη βελτιστοποίηση αφορά στη διατήρηση ενός παραθύρου στη μνήμη κατά το πρώτο στάδιο της διαδικασίας ταξινόμησης, όπου αποθηκεύονται αντίγραφα των στοιχείων με την καλύτερη εντροπία. Με βάση αυτά τα στοιχεία, μπορούν να απορριφθούν άμεσα τα στοιχεία που είναι βέβαιο ότι δεν ανήκουν στο skyline με αποτέλεσμα να μειωθούν οι απαιτούμενες συγκρίσεις. Η δεύτερη βελτιστοποίηση συνδυάζει το τελευταίο στάδιο της διαδικασίας ταξινόμησης με το πρώτο στάδιο του αλγορίθμου SFS, απορρίπτοντας και τα εναπομείναντα κυριαρχούμενα στοιχεία του συνόλου, εξάγοντας έτσι το τελικό skyline.

Ουσιαστικά, με την πρώτη βελτιστοποίηση επιτυγχάνεται η μείωση του μεγέθους του συνόλου δεδομένων που χρειάζεται να επεξεργαστεί ο αλγόριθμος, ενώ επιπλέον με τη δεύτερη βελτιστοποίηση εξασφαλίζεται η εξοικονόμηση μιας επανάληψης του αλγορίθμου.

Στη χειρότερη περίπτωση, η πολυπλοκότητα του αλγορίθμου LESS είναι $O(dn^2)$ όπου d είναι ο αριθμός των διαστάσεων και n είναι το μέγεθος του συνόλου δεδομένων.

3.10 Sort and Limit Skyline Algorithm

Στην προσπάθειά τους να ξεπεράσουν τα μειονεκτήματα των προαναφερθέντων αλγορίθμων, οι Bartolini et al. [42] πρότειναν τον αλγόριθμο Sort and Limit Skyline Algorithm (SaLSa), ο οποίος στοχεύει στην εύρεση του skyline με όσο το δυνατόν λιγότερες συγκρίσεις μεταξύ των στοιχείων του συνόλου δεδομένων.

Όπως οι αλγόριθμοι SFS και LESS, έτσι και ο SaLSa χρησιμοποιεί μια συνάρτηση ταξινόμησης των στοιχείων πριν από τον έλεγχο της κυριαρχίας τους. Η διαφορά είναι ότι σε αυτή την περίπτωση η συνάρτηση ταξινόμησης επιλέγεται με τέτοιο τρόπο ώστε να εγγυάται ότι τα στοιχεία από ένα σημείο και έπειτα είναι βέβαιο ότι κυριαρχούνται και δεν είναι αναγκαίο να διαβαστούν από τον αλγόριθμο.

Συγκεκριμένα, οι ερευνητές που πρότειναν τον αλγόριθμο χρησιμοποίησαν μια συνάρτηση ταξινόμησης με βάση τη μικρότερη συντεταγμένη $f\ min(p) = (min\, p[i], sum(p)),\ \ i \in [1, d]$ όπου $sum(p) = \sum_{i=1}^{d} p[i]$. Αν S είναι το σύνολο των στοιχείων του skyline μια δεδομένη στιγμή, τότε για κάθε στοιχείο $p \in S$ η μέγιστη τιμή από όλες τις διαστάσεις του είναι η $\boldsymbol{p} = \max\{p[i]\}$. Η τιμή που χρησιμοποιείται ως παράγοντας τερματισμού υποδεικνύοντας ότι όλες οι υπόλοιπες τιμές από το ταξινομημένο σύνολο δεδομένων κυριαρχούνται είναι η $p_{stop} = \arg\min\{\boldsymbol{p}\},\ \ p \in S$. Κάθε φορά που ένα νέο στοιχείο διαβάζεται από τον αλγόριθμο, συγκρίνεται με όλα τα στοιχεία που ανήκουν στο skyline κατά τη δεδομένη στιγμή. Αν κυριαρχείται, τότε απορρίπτεται, ειδάλλως προστίθεται στο skyline και ο αλγόριθμος ελέγχει τον παράγοντα τερματισμού. Αν η τιμή τερματισμού είναι μικρότερη ή ίση με τη την τιμή της συνάρτησης $f\ \min(p)$ του συγκεκριμένου στοιχείου, τότε ο αλγόριθμος τερματίζεται αφού είναι περιττό να εξεταστούν τα εναπομείναντα στοιχεία.

Ο αλγόριθμος SaLSa υπολογίζει προοδευτικά το skyline έχοντας παράλληλα το πλεονέκτημα της μείωσης των υπό εξέταση στοιχείων του συνόλου δεδομένων. Το παραπάνω, σε συνδυασμό και με τα πλεονεκτήματα των αλγορίθμων SFS και LESS στους οποίους βασίζεται, τον καθιστά ιδιαίτερα αποδοτικό.

3.11 Άλλοι Αλγόριθμοι

Ο αλγόριθμος OSPS (Object Space Partitioning Skyline) που προτάθηκε από τους Zhang et al. [43] και αποτελεί βελτίωση του DC, υπολογίζει προοδευτικά το skyline διαιρώντας αναδρομικά τον d-διάστατο χώρο σε 2^d επιμέρους τμήματα έχοντας ως οδηγό ένα στοιχείο του skyline. Με αυτό τον τρόπο, τα στοιχεία του skyline οργανώνονται σε ένα δέντρο αναζήτησης όπου κάθε στοιχείο

συγκρίνεται μόνο με ορισμένα από τα στοιχεία του τρέχοντος skyline. Το αποτέλεσμα είναι ότι μειώνεται δραστικά ο αριθμός των απαιτούμενων συγκρίσεων.

Ο αλγόριθμος BSkyTree [44], όπως και ο OSPS, διαιρεί αναδρομικά το σύνολο δεδομένων έχοντας κάθε φορά ως οδηγό ένα στοιχείο του skyline. Η διαφορά και η βελτίωση ως προς τον OSPS είναι ο τρόπος με τον οποίο επιλέγεται το στοιχείο. Σε αντίθεση με τον OSPS, ο BSkyTree δεν επιλέγει ένα τυχαίο στοιχείο του skyline αλλά το πλησιέστερο στην κύρια διαγώνιο. Η επιλογή αυτή είναι η βέλτιστη ως προς τη διαίρεση του συνόλου σε ισορροπημένα τμήματα, γεγονός το οποίο μειώνει ακόμα περισσότερο τον αριθμό των απαιτούμενων συγκρίσεων.

Μια ξεχωριστή κατηγορία αποτελούν οι αλγόριθμοι που εκμεταλλεύονται τους πολυπύρηνους επεξεργαστές, όπως για παράδειγμα ο APSkyline που παρουσιάστηκε από τους Liknes et al. [45] και ο Hybrid που προτάθηκε από τους Chester et al. [46]. Μια πλήρης ανασκόπηση των υπαρχόντων αλγορίθμων για πολυπύρηνους επεξεργαστές έχει γίνει από τους Khames et al. [47].

Μια άλλη κατηγορία αλγορίθμων έχουν σχεδιαστεί και υλοποιηθεί πάνω στους επεξεργαστές των καρτών γραφικών. Μερικοί από τους πιο διαδεδομένους είναι οι GNL (GPU-based Nested Loop) [48], GGS [49], και SkyAlign [50].

Επιπλέον, έχουν προταθεί πλήθος παράλληλων αλγορίθμων στη βάση των κατανεμημένων συστημάτων και του μοντέλου Map Reduce [51]–[55].

Όλοι οι προαναφερθέντες αλγόριθμοι έχουν σχεδιαστεί στη βάση στατικών και πλήρων αριθμητικών δεδομένων. Όμως, πολλά από τα πραγματικά σύνολα δεδομένων είναι δυναμικά και ορισμένες φορές ελλιπή και αβέβαια, ενώ δεν αποτελούνται πάντα από αριθμητικές τιμές. Για τον υπολογισμό του skyline σε δυναμικά σύνολα δεδομένων έχουν μελετηθεί και προταθεί αλγόριθμοι όπως για παράδειγμα αυτός των Dehaki et al. [56]. Αντίστοιχα, έχουν υλοποιηθεί αλγόριθμοι για ελλιπή και αβέβαια σύνολα δεδομένων [57], [58]. Για κατηγορικά δεδομένα, έχουν επίσης προταθεί ξεχωριστοί αλγόριθμοι [59], [60].

Τέλος, έχουν προταθεί μεθοδολογίες για τη μείωση της πολυπλοκότητας σε αυξημένο αριθμό διαστάσεων. Για παράδειγμα, οι Ghosh et al. [61] πρότειναν τη συγχώνευση των αλληλεξαρτώμενων χαρακτηριστικών του συνόλου δεδομένων σε ένα το οποίο θα αποτελεί συνάρτησή τους. Με αυτό τον τρόπο, συγχωνεύοντας πολλαπλά χαρακτηριστικά σε ένα, επιτυγχάνεται μείωση των συνολικών διαστάσεων του συνόλου δεδομένων και επομένως μείωση της πολυπλοκότητας.

Κεφάλαιο 4ο: Υλοποίηση τελεστή skyline στη γλώσσα Python

4.1 Η γλώσσα Python

Η Python είναι μια διερμηνευόμενη, υψηλού επιπέδου γλώσσα προγραμματισμού. Είναι εύκολη στην εκμάθηση και υποστηρίζει τόσο το διαδικαστικό προγραμματισμό όσο και τον αντικειμενοστρεφή αλλά και το συναρτησιακό. Μερικά από τα χαρακτηριστικά της είναι η φορητότητα (portability), η επεκτασιμότητα (extensibility) και η ενσωμάτωση (integration) με άλλες γλώσσες προγραμματισμού όπως C/C++/Java.

4.2 Δομοστοιχεία, πακέτα και βιβλιοθήκες

Ένα δομοστοιχείο (module) είναι ένα εκτελέσιμο αρχείο με επέκταση *.py* το οποίο περιέχει κλάσεις, συναρτήσεις και καθολικές μεταβλητές.

Πολλά δομοστοιχεία μαζί σε έναν κοινό κατάλογο αποτελούν ένα πακέτο (package). Ένα πακέτο μπορεί επίσης να περιέχει άλλα πακέτα.

Μια βιβλιοθήκη (library), όπως και ένα πακέτο, είναι μια συλλογή από πολλά δομοστοιχεία ή και πακέτα που καλύπτουν ένα ευρύ φάσμα λειτουργικότητας.

4.3 Δημιουργία και δημοσίευση βιβλιοθήκης Python

Στο ριζικό κατάλογο είναι απαραίτητο ένα αρχείο για το build της βιβλιοθήκης, συνήθως με το όνομα *setup.py*. Αυτό το αρχείο περιέχει πληροφορίες για τη βιβλιοθήκη όπως το όνομα, την έκδοση, τις εξαρτήσεις αλλά και πληροφορίες για το δημιουργό της.

Σε κάθε κατάλογο αλλά και υποκατάλογο της βιβλιοθήκης πρέπει να βρίσκεται ένα αρχείο με όνομα *__init__.py* με το οποίο ο διερμηνέας της Python αναγνωρίζει ότι ο κατάλογος πρόκειται για ένα πακέτο. Το αρχείο αυτό μπορεί να είναι κενό ή να περιέχει ονόματα μεθόδων που βρίσκονται στον εκάστοτε κατάλογο.

Όσο αφορά τη δημοσίευση της βιβλιοθήκης, το PyPi αποτελεί το επίσημο αποθετήριο για όλες τις βιβλιοθήκες της Python. Για να ανεβάσει κάποιος μια βιβλιοθήκη στο PyPi πρέπει να διαθέτει λογαριασμό. Για τη δημιουργία του τελικού πακέτου προς εγκατάσταση, με την εντολή *python setup.py sdist bdist_ wheel* δημιουργούνται οι κατάλογοι *dist, build* και *your_package.egg-info,* ενώ με το βοηθητικό εργαλείο *twine* και την εντολή *twine upload dist/** δημοσιεύονται στο PyPi τα περιεχόμενα του καταλόγου *dist*. Με το πέρας της διαδικασίας το πακέτο είναι πλέον δημόσια διαθέσιμο για εγκατάσταση μέσω της εντολής *pip install* ακολουθούμενης από το όνομα της βιβλιοθήκης.

4.4 Βελτιστοποίηση κώδικα Python για επιστημονικές εφαρμογές

Εξαιτίας της δυναμικής, διερμηνευόμενης φύσης της, η Python είναι πιο αργή στη διαχείριση πινάκων και την εφαρμογή τελεστών σε αυτούς σε σχέση με μια μεταγλωτιζώμενη γλώσσα όπως η C/C++. Για τη βελτίωση της απόδοσης σε αυτές τις περιπτώσεις, έχουν υλοποιηθεί βιβλιοθήκες όπως η NumPy, μεταγλωττιστές όπως οι Numba, Pythran και Nuitka, αλλά και γλώσσες που αποτελούν επέκταση της Python όπως η Cython. Στην παρούσα Π.Ε. γίνεται χρήση της βιβλιοθήκης NumPy και του μεταγλωττιστή Numba.

Η NumPy παρέχει ένα αντικείμενο πολυδιάστατου πίνακα μαζί με ένα πλήθος από μεθόδους για τη γρήγορη επεξεργασία των πινάκων μεταξύ των οποίων μαθηματικοί και λογικοί τελεστές, ταξινόμηση, επιλογή, αλλά και μέθοδοι γραμμικής άλγεβρας και στατιστικής.

Η υπεροχή που προσφέρει η NumPy ως προς την ταχύτητα, οφείλεται σε δύο χαρακτηριστικά της: τη διανυσματοποίηση και την αναμετάδοση. Με τον όρο διανυσματοποίηση περιγράφεται η απουσία βρόγχων και δεικτών στον κώδικα, η οποία τον καθιστά συνοπτικότερο και περισσότερο κατανοητό στην ανάγνωση καθώς ομοιάζει περισσότερο με τους παραδοσιακούς μαθηματικούς συμβολισμούς. Η αναμετάδοση αφορά στην ικανότητα της NumPy να διαχειρίζεται πίνακες διαφορετικού μεγέθους κατά την εφαρμογή μαθηματικών, λογικών, δυαδικών και λοιπών τελεστών.

Ο μεταγλωττιστής Numba είναι ένας μεταγλωττιστής JIT (Just-In-Time) ο οποίος μεταγλωττίζει τις συναρτήσεις Python σε γλώσσα μηχανής προσεγγίζοντας έτσι τις ταχύτητες εκτέλεσης γλωσσών όπως οι C και Fortran. Επιπλέον, παρέχει επιλογές για παραλληλισμό του κώδικα τόσο για CPU όσο και για GPU.

4.5 Ανάπτυξη βιβλιοθήκης Python για υπολογισμό skyline

Στόχος της βιβλιοθήκης που αναπτύχθηκε είναι ο υπολογισμός ενός ή περισσότερων επιπέδων skyline από ένα σύνολο δεδομένων με βάση επιλογές του χρήστη όπως ο τύπος της κυριαρχίας, το ζητούμενο για κάθε διάσταση και ο μέγιστος αριθμός skylines που επιθυμεί να υπολογίσει.

Συγκεκριμένα, η συνάρτηση που καλείται να εκτελέσει ο χρήστης δέχεται ως υποχρεωτική είσοδο τα παρακάτω:

- Σύνολο δεδομένων: Το σύνολο δεδομένων σε μορφή πίνακα numpy δύο διαστάσεων.
- Τύπος κυριαρχίας: Ο τύπος της κυριαρχίας σε μορφή συμβολοσειράς. Με την επιλογή «strict» εφαρμόζεται αυστηρή κυριαρχία ενώ σε οποιαδήποτε άλλη περίπτωση η προεπιλογή είναι η παραδοσιακή κυριαρχία.
- Ζητούμενα: Τα ζητούμενα για κάθε διάσταση σε μορφή πίνακα συμβολοσειρών μεγέθους ίσο με τον αριθμό των διαστάσεων. Με την επιλογή «max» δηλώνεται η επιθυμία για μεγιστοποίηση της αντίστοιχης διάστασης ενώ σε οποιαδήποτε άλλη περίπτωση η προεπιλογή είναι η ελαχιστοποίηση.
- Μέγιστο βάθος: Ο μέγιστος επιθυμητός αριθμός skylines σε μορφή ακεραίου. Για τον αριθμό μηδέν υπολογίζονται όλα τα skylines ενώ για οποιονδήποτε θετικό ακέραιο υπολογίζεται ο αντίστοιχος αριθμός skylines (ή ο μέγιστος αν τον ξεπερνάει).

Η έξοδος που παράγεται είναι ένας μονοδιάστατος πίνακας numpy από ακέραιους αριθμούς, μήκους ίσου με τον αριθμό των δεδομένων. Ο ακέραιος αριθμός της κάθε θέσης του πίνακα αντιπροσωπεύει τον αύξων αριθμό του skyline στο οποίο ανήκει το αντίστοιχο στοιχείο του συνόλου δεδομένων.

Ενδεικτική είσοδος και έξοδος της συνάρτησης φαίνεται στην παρακάτω εικόνα όπου υπολογίζονται όλα τα skylines του συνόλου δεδομένων με όνομα *my_dataset* με βάση την παραδοσιακή κυριαρχία για ελαχιστοποίηση της πρώτης και της δεύτερης διάστασης και μεγιστοποίηση της τρίτης.

```
>> skyline(my_dataset, 'normal', ['min', 'min', 'max'], 0)
[4 1 2 … 1 1 2]
```

Εικόνα 4.1: Εκτέλεση κύριας συνάρτησης της βιβλιοθήκης.

Για τον υπολογισμό των skylines επιλέχθηκε η υλοποίηση του αλγορίθμου BNL όπως αυτός παρουσιάστηκε στην Υποενότητα 3.2. Συγκεκριμένα, δεχόμενος ως είσοδο το σύνολο δεδομένων σε μορφή πίνακα numpy δύο διαστάσεων, υπολογίζει και επιστρέφει ως έξοδο έναν πίνακα numpy ίσου μήκους με το σύνολο δεδομένων και αποτελούμενο από μηδενικά και άσσους όπου οι άσσοι υποδεικνύουν τις θέσεις των στοιχείων του skyline. Για τον υπολογισμό περισσότερων των ενός skylines, εκτελείται επαναληπτικά ο αλγόριθμος BNL με είσοδο κάθε φορά τα στοιχεία του συνόλου που δεν ανήκουν σε ήδη υπολογισμένο skyline.

Όσο αφορά την κυριαρχία υλοποιήθηκαν δύο συναρτήσεις, μια για την παραδοσιακή κυριαρχία και μια για την αυστηρή όπως αυτές ορίστηκαν στην Υποενότητα 2.2.1. Σε κάθε περίπτωση, λήφθηκαν υπόψη οι πιθανές None τιμές σε ελλιπή σύνολα δεδομένων οι οποίες και αγνοούνται κατά τη σύγκριση. Οι συναρτήσεις αυτές υπολογίζουν την κυριαρχία μόνο με τους τελεστές $<$ και $\leq$, επομένως στις διαστάσεις όπου ζητείται μεγιστοποίηση οι τιμές μετατρέπονται στις αντίθετες με την έναρξη της εκτέλεσης του αλγορίθμου. Με αυτό τον τρόπο επιτυγχάνεται ομοιόμορφος και αποδοτικότερος υπολογισμός της κυριαρχίας στο σύνολο δεδομένων αφού αποφεύγεται ο έλεγχος του ζητούμενου της διάστασης σε κάθε υπολογισμό.

Κεφάλαιο 5ο: Πειραματική αξιολόγηση

5.1 Περιβάλλον εκτέλεσης

Η πειραματική αξιολόγηση της απόδοσης του αλγορίθμου έγινε σε περιβάλλον Windows 10 64-bit με τον επεξεργαστή AMD Ryzen 5 1600 και τη μνήμη RAM Corsair Vengeance LPX 2400. Ο παραπάνω επεξεργαστής αποτελείται από έξι πυρήνες, διαθέτει τρία επίπεδα μνήμης cache (576 KB, 3 MB και 16 MB) και ρολόι χρονισμένο στα 3.20 GHz. Αντίστοιχα, η μνήμη RAM έχει χωρητικότητα 16 GB και είναι χρονισμένη στα 2400 MHz.

Ο κώδικας εκτελέστηκε στο τερματικό Powershell του συστήματος.

5.2 Πειραματικά σύνολα δεδομένων

Χρησιμοποιήθηκαν 30 συνθετικά σύνολα δεδομένων και 2 πραγματικά σύνολα δεδομένων κλιμακούμενου μεγέθους τόσο ως προς τον αριθμό των δεδομένων όσο και ως προς τον αριθμό των χαρακτηριστικών τους. Συγκεκριμένα, χρησιμοποιήθηκαν ομοιόμορφα συνθετικά σύνολα των 100, 1000, 10000, 100000 και 1000000 στοιχείων με 2, 3, 4, 7, 10 και 15 διαστάσεις το καθένα. Τα πραγματικά σύνολα δεδομένων αποτελούνταν από 17264 και 127931 στοιχεία με 8 και 6 διαστάσεις αντίστοιχα [62].

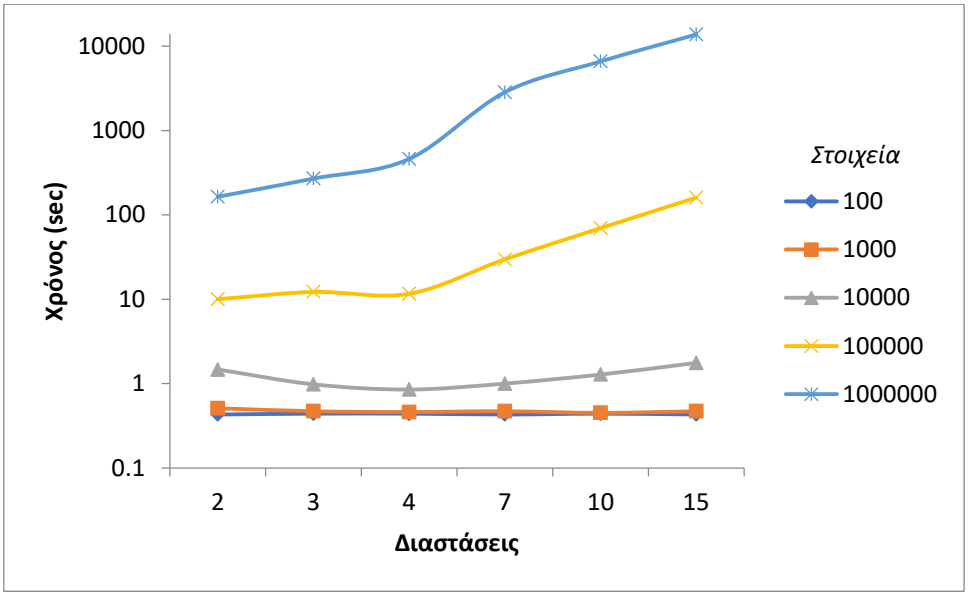
5.3 Αποτελέσματα εκτέλεσης

Οι χρόνοι εκτέλεσης του αλγορίθμου σε δευτερόλεπτα για τα συνθετικά σύνολα δεδομένων φαίνονται στον παρακάτω πίνακα.

Πίνακας 5.1. Χρόνος εκτέλεσης του αλγορίθμου σε δευτερόλεπτα για συνθετικά σύνολα δεδομένων.

Διαστάσεις / *Στοιχεία*	***2***	***3***	***4***	***7***	***10***	***15***
100	0.43	0.44	0.44	0.43	0.44	0.43
1000	0.51	0.47	0.46	0.47	0.45	0.47
10000	1.47	0.98	0.85	1.00	1.28	1.76
100000	10.00	12.20	11.60	29.70	69.60	160.00
1000000	164.00	269.00	460.00	2840.00	6585.00	13726.00

Ομοίως, τα αποτελέσματα φαίνονται και στο παρακάτω διάγραμμα.



Διάγραμμα 5.1. Χρόνος εκτέλεσης του αλγορίθμου σε δευτερόλεπτα για συνθετικά σύνολα δεδομένων.

Όπως είναι αναμενόμενο, με την αύξηση των στοιχείων και των διαστάσεων αυξάνεται και ο χρόνος υπολογισμού. Για μικρό αριθμό στοιχείων και διαστάσεων η αύξηση είναι αμελητέα, ενώ όσο αυξάνονται τα στοιχεία η επιβάρυνση είναι αισθητή.

Παρατηρείται, επίσης, ότι στα μικρότερα σύνολα δεδομένων η προσθήκη μερικών διαστάσεων μειώνει ελάχιστα το χρόνο υπολογισμού. Το φαινόμενο αυτό πιθανώς σχετίζεται με την «κατάρα των διαστάσεων» που περιγράφηκε στην υποενότητα 2.2.3. Συγκεκριμένα, όσο αυξάνονται οι διαστάσεις τόσο αυξάνεται η πιθανότητα ένα στοιχείο να ανήκει στο πρώτο skyline. Συνεπώς, τα συνολικά skylines ενός συνόλου με περισσότερες διαστάσεις έχουν περισσότερες πιθανότητες να είναι μικρότερα σε αριθμό σε σχέση με ένα σύνολο ίδιου μεγέθους αλλά λιγότερων διαστάσεων. Σε μια τέτοια περίπτωση απαιτούνται λιγότερες συγκρίσεις και υπολογισμοί, επομένως ο συνολικός χρόνος εκτέλεσης του αλγορίθμου είναι μικρότερος.

Οι χρόνοι εκτέλεσης του αλγορίθμου σε δευτερόλεπτα για τα πραγματικά σύνολα δεδομένων φαίνονται στον παρακάτω πίνακα.

Πίνακας 5.2. Χρόνος εκτέλεσης του αλγορίθμου σε δευτερόλεπτα για πραγματικά σύνολα δεδομένων

Διαστάσεις / *Στοιχεία*	***6***	***8***
17264		1.90
127931	25.00	

Τα παραπάνω αποτελέσματα φαίνεται να είναι σύμφωνα με τα αποτελέσματα των συνθετικών δεδομένων, στοιχείο το οποίο δείχνει ότι πρόκειται για αρκετά ομοιόμορφα δεδομένα.

Κεφάλαιο 6ο: Συμπεράσματα και προτάσεις για βελτίωση

Ο υπολογισμός του skyline αποτελεί σημαντικό εργαλείο για τη λήψη πολυκριτηριακών αποφάσεων. Η συνεχής αύξηση του όγκου των διαθέσιμων δεδομένων συνιστά πρόκληση στην εκτέλεση υπολογισμών πάνω σε μεγάλα σύνολα δεδομένων, ιδιαίτερα συνυπολογίζοντας και την ύπαρξη ελλιπών δεδομένων. Πλήθος αλγορίθμων έχουν προταθεί για τον αποδοτικό υπολογισμό του skyline, με μερικούς από αυτούς να είναι ειδικά σχεδιασμένοι για ιδιαίτερες περιπτώσεις συνόλων δεδομένων όπως ελλιπή ή κατηγορικά. Στην παρούσα Π.Ε. μελετήθηκαν οι βασικές έννοιες που σχετίζονται με τον τελεστή skyline όπως επίσης και οι κυριότεροι αλγόριθμοι που έχουν προταθεί για τον υπολογισμό του. Ένας από αυτούς τους αλγόριθμους χρησιμοποιήθηκε για την υλοποίηση μιας βιβλιοθήκης python με στόχο τον επαναληπτικό υπολογισμό όλων των skylines ενός δεδομένου συνόλου δεδομένων.

Τα αποτελέσματα των πειραματικών εκτελέσεων ήταν σε πλήρη συμφωνία με τη θεωρία, δηλαδή με την αύξηση των στοιχείων και των διαστάσεων αυξήθηκε και ο χρόνος υπολογισμού των skylines. Σε μικρά σύνολα δεδομένων, μικρή αύξηση στον αριθμό των διαστάσεων δεν επέφερε αντίστοιχη αύξηση στο χρόνο υπολογισμού αλλά μικρή μείωση. Αυτό το φαινόμενο ερμηνεύεται ως αποτέλεσμα της «κατάρας των διαστάσεων» όπου αύξηση των διαστάσεων ενός συνόλου συνεπάγεται αύξηση των στοιχείων που ανήκουν στο πρώτο skyline και επομένως λιγότερες συγκρίσεις στοιχείων κατά τον υπολογισμό όλων των skylines.

Μελλοντικά, η βιβλιοθήκη μπορεί να επεκταθεί με αποδοτικότερους αλγορίθμους τόσο για CPU όσο και για GPU, αλλά και με αλγορίθμους για κατανεμημένα συστήματα. Επίσης, η βιβλιοθήκη θα μπορούσε να επεκταθεί και με αλγορίθμους για ιδιαίτερα σύνολα δεδομένων όπως για σύνολα με κατηγορικά δεδομένα.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] S. Borzsony, D. Kossmann, and K. Stocker, “The Skyline operator,” in *Proceedings 17th International Conference on Data Engineering*, Heidelberg, Germany, 2001, pp. 421–430.

[2] B. Zhang, S. Zhou, and J. Guan, “Adapting Skyline Computation to the MapReduce Framework: Algorithms and Experiments,” in *Database Systems for Adanced Applications*, Berlin, Heidelberg, 2011, pp. 403–414.

[3] H.-K. Hwang, T.-H. Tsai, and W.-M. Chen, “Threshold Phenomena in k-Dominant Skylines of Random Samples,” *SIAM J. Comput.*, vol. 42, no. 2, pp. 405–441, Jan. 2013.

[4] Y.-W. Peng and W.-M. Chen, “Parallel k-dominant skyline queries in high-dimensional datasets,” *Information Sciences*, vol. 496, pp. 538–552, Sep. 2019.

[5] H. T. Kung, F. Luccio, and F. P. Preparata, “On Finding the Maxima of a Set of Vectors,” *J. ACM*, vol. 22, no. 4, pp. 469–476, Oct. 1975.

[6] P. Godfrey, R. Shipley, and J. Gryz, “Maximal Vector Computation in Large Data Sets,” *VLDB*, pp. 229–240, 2005.

[7] V. Pareto, *Cours d’économie politique*. Librairie Droz, 1964.

[8] S. Kishida, S. Ueda, A. Keyaki, and J. Miyazaki, “Skyline-Based Recommendation Considering User Preferences,” in *Web and Big Data*, Cham, 2017, pp. 133–141.

[9] F. E. Bousnina, S. Elmi, M. Chebbah, M. A. Bach Tobji, A. HadjAli, and B. Ben Yaghlane, “Skyline Operator over Tripadvisor Reviews Within the Belief Functions Framework,” in *Digital Economy. Emerging Technologies and Business Innovation*, Cham, 2017, pp. 186–197.

[10] M. Che, L. Wang, and Z. Jiang, “An Approach to Multidimensional Medical Data Analysis Based on the Skyline Operator,” in *2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, Dec. 2018, pp. 1806–1810.

[11] A. Mutlu, F. Goz, K. Koksal, and A. Erener, “Landslide Susceptibility Assessment using Skyline Operator and Majority Voting,” *Sakarya University Journal of Science*, vol. 23, no. 5, pp. 782–787, Oct. 2019.

[12] A. Alem, Y. Dahmani, and B. Mebarek, “Skyline Computation for Improving Naïve Bayesian Classifier in Intrusion Detection System,” *ISI*, vol. 24, no. 5, pp. 513–518, Nov. 2019.

[13] R. M. Alguliyev, R. M. Aliguliyev, R. G. Alakbarov, and O. R. Alakbarov, “The Skyline Operator for Selection of Virtual Machines in Mobile Computing,” *IJMECS*, vol. 10, no. 11, pp. 1–10, Nov. 2018.

[14] I. Kertiou *et al.*, “A dynamic skyline technique for a context-aware selection of the best sensors in an IoT architecture,” *Ad Hoc Networks*, vol. 81, pp. 183–196, Dec. 2018.

[15] J. Li, Y. Yan, and D. Lemire, “Scaling Up Web Service Composition with the Skyline Operator,” in *2016 IEEE International Conference on Web Services (ICWS)*, Jun. 2016, pp. 147–154.

[16] A. Ouadah, A. Hadjali, F. Nader, and K. Benouaret, “SEFAP: an efficient approach for ranking skyline web services,” *J Ambient Intell Human Comput*, vol. 10, no. 2, pp. 709–725, Feb. 2019.

[17] S. Wang, Y. Li, Z. Li, J. Wu, and F. Chen, “Surveillance Methods of Running Condition of Chemical Equipments Based on Skyline,” in *2016 IEEE Trustcom/BigDataSE/ISPA*, Aug. 2016, pp. 2246–2250.

[18] L. G. Asri and Annisa, “Application of Skyline Query on Route Selection (the Case Study of Bogor City Roadway),” in *2020 International Conference on Computer Science and Its Application in Agriculture (ICOSICA)*, Sep. 2020, pp. 1–6.

[19] Q. Wan, R. C.-W. Wong, I. F. Ilyas, M. T. Özsu, and Y. Peng, “Creating competitive products,” *Proc. VLDB Endow.*, vol. 2, no. 1, pp. 898–909, Aug. 2009.

[20] X. Zhou, K. Li, Z. Yang, and K. Li, “Finding Optimal Skyline Product Combinations under Price Promotion,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 1, pp. 138–151, Jan. 2019.

[21] A. Sidiropoulos, A. Gogoglou, D. Katsaros, and Y. Manolopoulos, “Gazing at the skyline for star scientists,” *Journal of Informetrics*, vol. 10, no. 3, pp. 789–813, Aug. 2016.

[22] G. Stoupas, A. Sidiropoulos, D. Katsaros, and Y. Manolopoulos, “Skyline-Based University Rankings,” in *ADBIS, TPDL and EDA 2020 Common Workshops and Doctoral Consortium*, Cham, 2020, pp. 347–352.
[23] J. Zheng and S. Zhang, “Adding ReputationRank to member promotion using skyline operator in social networks,” *Comput Soc Netw*, vol. 5, no. 1, p. 7, Sep. 2018.
[24] A. Abidi, S. Elmi, M. A. Bach Tobji, A. HadjAli, and B. Ben Yaghlane, “Skyline queries over possibilistic RDF data,” *International Journal of Approximate Reasoning*, vol. 93, pp. 277–289, Feb. 2018.
[25] W. Dhifli, N. E. I. Karabadji, and M. Elati, “Evolutionary mining of skyline clusters of attributed graph data,” *Information Sciences*, vol. 509, pp. 501–514, Jan. 2020.
[26] B. Bouderar, L. Alaoui, and M. Y. Hadi, “Solving the Vehicle Routing Problem using Skyline,” in *Proceedings of the New Challenges in Data Sciences: Acts of the Second Conference of the Moroccan Classification Society*, New York, NY, USA, Mar. 2019, pp. 1–4.
[27] M. L. Yiu and N. Mamoulis, “Efficient Processing of Top-k Dominating Queries on Multi-Dimensional Data,” in *Proc. 33rd Int’l Conf. Very Large Data Bases (VLDB)*, 2007, pp. 483–494.
[28] D. Papadias, Y. Tao, G. Fu, and B. Seeger, “An optimal and progressive algorithm for skyline queries,” in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, New York, NY, USA, Jun. 2003, pp. 467–478.
[29] C.-Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang, “Finding k-dominant skylines in high dimensional space,” in *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, New York, NY, USA, Jun. 2006, pp. 503–514.
[30] Y. Yuan, X. Lin, Q. Liu, W. Wang, J. X. Yu, and Q. Zhang, “Efficient Computation of the Skyline Cube,” in *Proceedings of the 31st international conference on Very large data bases*, 2005, pp. 241–252.
[31] J. Pei *et al.*, “Towards multidimensional subspace skyline analysis,” *ACM Trans. Database Syst.*, vol. 31, no. 4, pp. 1335–1381, Dec. 2006.
[32] N. Hanusse, P. Kamnang Wanko, and S. Maabout, “Computing and Summarizing the Negative Skycube,” in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, New York, NY, USA, Oct. 2016, pp. 1733–1742.
[33] K. Alami, N. Hanusse, P. Kamnang-Wanko, and S. Maabout, “The negative skycube,” *Information Systems*, vol. 88, p. 101443, Feb. 2020.
[34] J. Pei, B. Jiang, X. Lin, and Y. Yuan, “Probabilistic Skylines on Uncertain Data,” in *Proceedings of the 33rd international conference on Very large data bases*, 2007, pp. 15–26.
[35] E. Dellis and B. Seeger, “Efficient Computation of Reverse Skyline Queries,” *VLDB*, vol. 7, pp. 291–302.
[36] J. Liu, L. Xiong, J. Pei, J. Luo, and H. Zhang, “Finding Pareto optimal groups: group-based skyline,” *Proc. VLDB Endow.*, vol. 8, no. 13, pp. 2086–2097, Sep. 2015.
[37] H. Zhu, X. Li, Q. Liu, and H. Zhu, “Computing skyline groups: an experimental evaluation,” *Tsinghua Science and Technology*, vol. 24, no. 2, pp. 171–182, Apr. 2019.
[38] M. Sharifzadeh and C. Shahabi, “The Spatial Skyline Queries,” in *Proceedings of the 32nd international conference on Very large data bases*, 2006, pp. 751–762.
[39] K.-L. Tan, P.-K. Eng, and B. C. Ooi, “Efficient progressive skyline computation,” in *Proceedings of the 27th International Conference on Very Large Data Bases*, 2001, vol. 1, pp. 301–310.
[40] D. Kossmann, F. Ramsak, and S. Rost, “Chapter 25 - Shooting Stars in the Sky: An Online Algorithm for Skyline Queries,” in *VLDB ’02: Proceedings of the 28th International Conference on Very Large Databases*, P. A. Bernstein, Y. E. Ioannidis, R. Ramakrishnan, and D. Papadias, Eds. San Francisco: Morgan Kaufmann, 2002, pp. 275–286.
[41] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, “Skyline with Presorting: Theory and Optimizations,” in *Intelligent Information Processing and Web Mining*, Berlin, Heidelberg, 2005, pp. 595–604.
[42] I. Bartolini, P. Ciaccia, and M. Patella, “SaLSa: computing the skyline without scanning the whole sky,” in *Proceedings of the 15th ACM international conference on Information and knowledge management*, New York, NY, USA, Nov. 2006, pp. 405–414.

[43] S. Zhang, N. Mamoulis, and D. W. Cheung, “Scalable skyline computation using object-based space partitioning,” in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, New York, NY, USA, Jun. 2009, pp. 483–494.
[44] J. Lee and S. Hwang, “BSkyTree: scalable skyline computation using a balanced pivot selection,” in *Proceedings of the 13th International Conference on Extending Database Technology*, New York, NY, USA, Mar. 2010, pp. 195–206.
[45] S. Liknes, A. Vlachou, C. Doulkeridis, and K. Nørvåg, “APSkyline: Improved Skyline Computation for Multicore Architectures,” in *Database Systems for Advanced Applications*, Cham, 2014, pp. 312–326.
[46] S. Chester, D. Šidlauskas, I. Assent, and K. S. Bøgh, “Scalable parallelization of skyline computation for multi-core processors,” in *2015 IEEE 31st International Conference on Data Engineering*, Apr. 2015, pp. 1083–1094.
[47] W. Khames, A. Hadjali, and M. Lagha, “Skyline Computation on Multicore Architectures: A Survey,” in *2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI)*, Oct. 2020, pp. 1–6.
[48] W. Choi, L. Liu, and B. Yu, “Multi-criteria decision making with skyline computation,” in *2012 IEEE 13th International Conference on Information Reuse Integration (IRI)*, Aug. 2012, pp. 316–323.
[49] K. S. Bøgh, I. Assent, and M. Magnani, “Efficient GPU-based skyline computation,” in *Proceedings of the Ninth International Workshop on Data Management on New Hardware*, New York, NY, USA, Jun. 2013, pp. 1–6.
[50] K. S. Bøgh, S. Chester, and I. Assent, “SkyAlign: a portable, work-efficient skyline algorithm for multicore and GPU architectures,” *The VLDB Journal*, vol. 25, no. 6, pp. 817–841, Dec. 2016.
[51] Y. Park, J. Min, and K. Shim, “Efficient Processing of Skyline Queries Using MapReduce,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 5, pp. 1031–1044, May 2017.
[52] B. Bouderar, L. Alaoui, and M. Y. Hadi, “A Parallel Nearest Neighbor Algorithm for Skyline Computation Using Map Reduce,” in *Big Data and Smart Digital Environment*, Cham, 2019, pp. 204–214.
[53] C. Kalyvas and M. Maragoudakis, “Skyline and reverse skyline query processing in SpatialHadoop,” *Data & Knowledge Engineering*, vol. 122, pp. 55–80, Jul. 2019.
[54] H. Wijayanto, W. Wang, W. Ku, and A. Chen, “LShape Partitioning: Parallel Skyline Query Processing using MapReduce,” *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2020.
[55] E. Gavagsaz, “Parallel computation of probabilistic skyline queries using MapReduce,” *J Supercomput*, vol. 77, no. 1, pp. 418–444, Jan. 2021.
[56] G. B. Dehaki, H. Ibrahim, F. Sidi, N. I. Udzir, and A. A. Alwan, “A Rule-based Skyline Computation over a Dynamic Database,” in *Proceedings of the 22nd International Conference on Information Integration and Web-based Applications & Services*, New York, NY, USA, Nov. 2020, pp. 97–103.
[57] Y. Zeng, G. Chen, K. Li, Y. Zhou, X. Zhou, and K. Li, “M-Skyline: Taking sunk cost and alternative recommendation in consideration for skyline query on uncertain data,” *Knowledge-Based Systems*, vol. 163, pp. 204–213, Jan. 2019.
[58] S. Elmi and K.-L. Tan, “Efficient Skyline Computation over Incomplete and Uncertain Data for Decision Making Systems,” in *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, Nov. 2020, pp. 1059–1064.
[59] W. Lee, J. J. Song, and C. K.-S. Leung, “Categorical Data Skyline Using Classification Tree,” in *Web Technologies and Applications*, Berlin, Heidelberg, 2011, pp. 181–187.
[60] M. F. Rahman, A. Asudeh, N. Koudas, and G. Das, “Efficient Computation of Subspace Skyline over Categorical Domains,” in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, New York, NY, USA, Nov. 2017, pp. 407–416.
[61] P. Ghosh, L. J. Ghosh, S. Guha, N. C. Debnath, and S. Sen, “Reducing Computational Complexity of Skyline by the Use of Logical and Physical Bucket,” in *Contemporary Advances in Innovative and Applicable Information Technology*, Singapore, 2019, pp. 123–131.

[62] J. Lee and S. Hwang, “Scalable skyline computation using a balanced pivot selection technique,” *Information Systems*, vol. 39, pp. 1–21, Jan. 2014.