ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Ευφυής Διαχείριση Δεδομένων Μεγάλου Όγκου

Συγγραφέας
Χρήστος Καλύβας-Κασοπατίδης

Επιβλέπων
Μανώλης Μαραγκουδάκης

ΔΙΑΤΡΙΒΗ
για την απόκτηση Διδακτορικού Διπλώματος στο Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων, Πανεπιστήμιο Αιγαίου

Σάμος, Οκτώβριος, 2020

UNIVERSITY OF THE AEGEAN
SCHOOL OF ENGINEERING
Department of Information and Communication Systems Engineering

DOCTORAL THESIS

Intelligent Big Data Management

Author
Christos Kalyvas-Kasopatidis

Supervisor
Manolis Maragoudakis

A thesis submitted in Total Fulfilment of the Requirements for the Degree of Doctor of Philosophy (Ph.D.) at the Department of Information and Communication Systems Engineering, University of the Aegean

Samos, October 2020

Υπεύθυνη Δήλωση

Εγώ ο Χρήστος Καλύβας-Κασοπατίδης δηλώνω ότι είμαι ο αποκλειστικός συγγραφέας της υποβληθείσας διδακτορικής διατριβής με τίτλο «Ευφυής Διαχείριση Δεδομένων Μεγάλου Όγκου». Η συγκεκριμένη διδακτορική διατριβή είναι πρωτότυπη και εκπονήθηκε αποκλειστικά για την απόκτηση του διδακτορικού διπλώματος του Τμήματος Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων. Κάθε βοήθεια, την οποία είχα για την προετοιμασία της, αναγνωρίζεται πλήρως και αναφέρεται επακριβώς στην εργασία.

Επίσης, επακριβώς αναφέρω στην εργασία τις πηγές, τις οποίες χρησιμοποίησα, και μνημονεύω επώνυμά τα δεδομένα ή τις ιδέες που αποτελούν προϊόν πνευματικής ιδιοκτησίας άλλων, ακόμη κι εάν η συμπερίληψη τους στην παρούσα εργασία υπήρξε έμμεση ή παραφρασμένη. Γενικότερα, βεβαιώνω ότι κατά την εκπόνηση της διδακτορικής διατριβής έχω τηρήσει απαρέγκλιτα όσα ο νόμος ορίζει περί διανοητικής ιδιοκτησίας και έχω συμμορφωθεί πλήρως με τα προβλεπόμενα στο νόμο περί προστασίας προσωπικών δεδομένων και τις αρχές Ακαδημαϊκής δεοντολογίας.

Υπογραφή:

_____

Ημερομηνία: Οκτώβριος, 2020

_____

Declaration of Authorship

I, Christos Kalyvas-Kasopatidis, declare that this Thesis entitled, “Intelligent Big Data Management” and the work presented in it are my own. I confirm that:

- This work was done wholly while in candidature for a research degree at this University.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this Thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the Thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

_____

Date: October, 2020

_____

Advising Committee of this Doctoral Thesis

Professor Manolis Maragoudakis (Supervisor)
Department of Informatics
Ionian University, Greece

Professor Efstathios Stamatatos (Advisor)
Department of Information and Communication Systems Engineering
University of the Aegean, Greece

Assistant Professor Panagiotis Rizomiliotis (Advisor)
Department of Informatics and Telematics
Harokopio University, Greece

University of the Aegean, Greece

2020

Approved by the Examining Committee

Manolis Maragoudakis
Professor, Ionian University, Greece

Efstathios Stamatatos
Professor, University of the Aegean, Greece

Panagiotis Rizomiliotis
Assistant Professor, Harokopio University, Greece

Demosthenes Vouyioukas
Professor, University of the Aegean, Greece

Spyros Sioutas
Professor, University of Patras, Greece

Phivos Mylonas
Associate Professor, Ionian University, Greece

Katia Lida Kermanidis
Associate Professor, Ionian University, Greece

University of the Aegean, Greece

2020

Christos Kalyvas-Kasopatidis

Department of information and communication systems engineering
School of engineering
University of the Aegean

ABSTRACT

Department of Information and Communication Systems Engineering

School of Engineering

University of the Aegean

Doctor of Philosophy

by Christos Kalyvas-Kasopatidis

Rapidly evolving technologies are constantly expanding the need for analysis and utilization of existing data. Many organizations base their business viability on the analysis of market data as well as the data they produce either by exporting inherent useful statistics and performance indicators or by using them in the decision-making processes, where one of the most important parameters in their analysis is the parameter of time. To store and analyze the huge volume of data, new methods of data management and analysis are created. This was especially noticeable with the advent of Big Data. The technologies that were developed gave the opportunity to expand the methods that existed for conventional data but also to create new methods, techniques and systems so that they can provide the same or even better analytics. However, as technology advances with the advent of IoT, the volume of data and the number of data flows are increasing rapidly. These flows should be stored, analyzed and combined with other data to extract useful information. With the advent of ML / AI, more and more processes can be automated to generate new knowledge. One of the main problems, however, is the lack of marked data.

One of the most common queries performed to retrieve information from data are the skyline queries. The skyline queries belong to the category of multi-objective optimization problems and aim to retrieve a set of answers that meets some usually conflicting criteria. Using such queries is always helpful as it has many areas of application and can be very helpful in the decision-making process, where there are multiple criteria for achieving a goal and an optimal solution may not be unique. So far, the literature in this field of research shows a significant number of works is mainly concerned with conventional data and there is room for research in the field of Big Data.

Taking into account all the above, this Thesis aims to carry out an extensive review in the field of skyline queries, the detection of specifications and needs in data of an information system for maritime environments, the analysis of the time parameter in skyline queries, the development of skyline queries on tree structures specifically designed for Big Data and the implementation of a classifier specifically designed for Big Data environments.

More specifically, the first contribution is an extensive review of the existing work on skyline queries in which the skyline family is presented with a wide number of variations over the initial skyline query algorithm, the difference between index based and non-index-based methods and the applications that skyline queries have for problem solving. This review shows that skyline queries have evolved and allows readers to find areas that can be further explored.

The second contribution explores the various aspects of data in the context of a maritime information system. This analysis reviews the existing research area and the data needed to implement a maritime information system as well as the limitations that exist in processing and distributing the data. Through this research, the concept of Big Data became apparent, large data sets that are available for analysis were detected and was made clear that time parameterization is very important for performing data analytics.

The third contribution studied how can the dimension of time be integrated in skyline queries. The time dimension is an important parameter in data analysis and queries processing that is in many

cases is overlooked. This research reveals that the time parameter can affect the skyline, which shows that a special analysis needs to be made regarding the time dimension and to properly modification of the skyline queries in order to integrate the time dimension in them.

The fourth contribution examines the application of skyline queries in the field of Big Data and specifically SpatialHadoop. SpatialHadoop is an extension of the conventional Hadoop, which tries to integrate known tree structures that exist for conventional data in Hadoop. Through this analysis we can see the behavior of both types of skyline algorithms, that are indexed-based (or not) in Big Data environments and how the hybrid combinations work using skyline algorithms that are not based on an index over the indexed dataset created by the SpatialHadoop.

Finally, one of the biggest problems in deploying a machine learning model is the lack of labeled data. This lack is even more noticeable in Big Data environments as it is more difficult to point them out due their large volume. In the literature there are many mechanisms for labeling data depending on their application but there are no mechanisms for the efficient labeling of large volumes of data. Thus, in the fifth contribution, a classifier was created based on skyline questions. The use of skyline allows the creation of decision boundaries consisting of a small number of points.

Keywords: Skyline, Optimization, Temporal Skyline, Reverse Skyline, GIS, Maritime Data Technology and Applications, MapReduce, SpatialHadoop, Big Data, Classification, Decision Boundary

GREEK ABSTRACT

(Εκτεταμένη Περίληψη)

Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

Πολυτεχνική Σχολή

Πανεπιστήμιο Αιγαίου

Διδακτορική Διατριβή

Του Χρήστου Καλύβα-Κασοπατίδη

Οι ραγδαία αναπτυσσόμενες τεχνολογίες δημιουργούν ολοένα και μεγαλύτερες ανάγκες για την ανάλυση και την αξιοποίηση των υφιστάμενων δεδομένων. Πολλοί οργανισμοί βασίζουν την βιωσιμότητα τους στην ανάλυση των δεδομένων της αγοράς αλλά και των δεδομένων που παράγουν οι ίδιοι είτε μέσω της εξαγωγής χρήσιμων στατιστικών και δεικτών απόδοσης είτε αξιοποιώντας τα κατά τη διαδικασία λήψης αποφάσεων όπου μια από τις σημαντικότερες παραμέτρους στην ανάλυση τους είναι η παράμετρος του χρόνου. Για να μπορέσει να αποθηκευτεί και να αναλυθεί ο πολλές φορές τεράστιος όγκος δεδομένων δημιουργήθηκαν νέοι μέθοδοι διαχείρισης και ανάλυσης δεδομένων. Αυτό έγινε ιδιαίτερα αισθητό με την έλευση των Big Data. Οι τεχνολογίες που αναπτύχθηκαν έδωσαν την ευκαιρία της επέκτασης των μεθόδων που υπήρχαν για τα συμβατικά δεδομένα αλλά και την δημιουργία νέων μεθόδων, τεχνικών και συστημάτων ώστε να μπορούν να παρέχουν την ίδια ή ακόμα και καλύτερη ανάλυση. Καθώς όμως η τεχνολογία προχωράει με την έλευση του IOT ο όγκος των δεδομένων αλλά και οι ροές δεδομένων αυξάνονται ραγδαία. Οι ροές αυτές θα πρέπει να αποθηκευτούν να αναλυθούν και να συνδυαστούν με άλλα δεδομένα ώστε να εξαχθούν χρήσιμες πληροφορίες. Με την έλευση του ML/AI ολοένα και περισσότερα διαδικασίες μπορούν να αυτοματοποιηθούν παράγοντας αυτόματα καινούργια γνώση. Ένα από το κυριότερα προβλήματα που υπάρχουν όμως είναι η έλλειψη επισημασμένων δεδομένων.

Ένα από τα πιο διαδεδομένα ερωτήματα που υπάρχουν για την εξαγωγή συμπερασμάτων από τα δεδομένα είναι τα ερωτήματα κορυφογραμμής. Τα ερωτήματα κορυφογραμμής ανήκουν στην κατηγορία των multi-objective optimization προβλημάτων και έχουν ως στόχο την ανάκτηση ενός συνόλου απαντήσεων που ικανοποιεί κάποια συνήθως αντικρουόμενα κριτήρια. Η χρήση τέτοιου τύπου ερωτημάτων είναι πάντα χρήσιμη καθώς έχει πολλά πεδία εφαρμογής και μπορεί να βοηθήσει ιδιαίτερα στην διαδικασία λήψης αποφάσεων όπου υπάρχουν πολλαπλά κριτήρια για την επίτευξή ενός στόχου και η βέλτιστη λύση μπορεί να μην είναι μοναδική. Μέχρι στιγμής η βιβλιογραφία στο συγκεκριμένο ερευνητικό πεδίο εμφανίζει ένα σημαντικό πλήθος εργασιών οι οποίες κατά κύριο λόγο ασχολούνται με συμβατικά δεδομένα και υπάρχει χώρος για έρευνα στο πεδίο των Big Data.

Λαμβάνοντας υπόψη όλα τα παραπάνω η διατριβή αυτή έχει ως στόχο την πραγματοποίηση μιας εκτενούς ανασκόπησης στον χώρο των ερωτημάτων κορυφογραμμής, την ανίχνευση των προδιαγραφών και των αναγκών σε δεδομένα ενός πληροφοριακού συστήματος για θαλάσσιο περιβάλλον, την ανάλυση της παραμέτρου του χρόνου στα ερωτήματα κορυφογραμμής, την ανάπτυξη ερωτημάτων κορυφογραμμής σε δενδρικές δομές ειδικά σχεδιασμένες για Big Data και την δημιουργία ενός ταξινομητή (classifier) για μεγάλα δεδομένα.

Πιο αναλυτικά η πρώτη συνεισφορά είναι μια εκτενής ανασκόπηση του χώρου των ερωτημάτων κορυφογραμμής όπου θα παρουσιαστεί η οικογένεια των ερωτημάτων κορυφογραμμής με όλες τις παραλλαγές τους, την διαφοροποίηση ανάμεσα στις μεθόδους που βασίζονται η όχι σε ευρετήριο καθώς και τις εφαρμογές που έχουν τα ερωτήματα κορυφογραμμής για την επίλυση πληθώρας προβλημάτων. Μέσα από αυτή της ανασκόπηση παρουσιάζεται πως τα ερωτήματα

κορυφογραμμής εξελίχθηκαν και ανοίγει ο δρόμος για την εύρεση τομέων οι οποίοι μπορούν να διερευνηθούν περαιτέρω.

Στην δεύτερη συνεισφορά θα δούμε τις διάφορες πτυχές των δεδομένων στο πλαίσιο ενός θαλάσσιου πληροφοριακού συστήματος. Η ανάλυση που έγινε αφορούσε την ανασκόπηση του χώρου και των δεδομένων που χρειάζονται για την υλοποίηση ενός θαλάσσιου πληροφοριακού συστήματος καθώς και τους περιορισμούς που υπάρχουν στην επεξεργασία και την διακίνηση των δεδομένων αυτών. Μέσω της συγκεκριμένης έρευνας έγινε φανερή η έννοια των Big Data, ανιχνευθήκαν μεγάλα σύνολα δεδομένων τα οποία είναι διαθέσιμα για ανάλυση και είδαμε ότι η παράμετρος του χρόνου είναι πολύ σημαντική για την πραγματοποίηση αναλύσεων στα δεδομένα. Επίσης είδαμε του βασικότερους περιορισμούς στην διακίνηση και επεξεργασία των δεδομένων.

Στην τρίτη συνεισφορά μελετάτε ο τρόπος με τον οποίο μπορεί να ενσωματωθεί η διάσταση του χρόνου στα ερωτήματα κορυφογραμμής. Η διάσταση του χρόνου είναι μια σημαντική παράμετρος στην ανάλυση των δεδομένων και στην πραγματοποίηση επερωτήσεων η οποία πολλές φορές δεν λαμβάνεται υπόψη. Με αυτήν της έρευνα θα δούμε ότι η παράμετρος του χρόνου μπορεί να επηρεάσει τα αποτελέσματα ενός ερωτήματος κορυφογραμμής κάτι που καταδεικνύει πως χρειάζεται να γίνει ιδιαίτερη ανάλυση ως προς την διάσταση του χρόνου και να παραμετροποιηθούν κατάλληλα το ερώτημα κορυφογραμμής ώστε να ενσωματωθεί η διάσταση του χρόνου σε αυτά.

Η τέταρτη συνεισφορά εξετάζει την εφαρμογή των ερωτημάτων κορυφογραμμής στον χώρο των Big Data και συγκεκριμένα του SpatialHadoop. Το SpatialHadoop είναι μια επέκταση του συμβατικού Hadoop το οποίο προσπαθεί να ενσωματώσει τις δενδρικές δομές που υπάρχουν για τα συμβατικά δεδομένα στο Hadoop. Μέσω αυτής της ανάλυσης μπορούμε να δούμε την συμπεριφορά των αλγορίθμων κορυφογραμμής που δεν χρησιμοποιούν κάποια ευρετηρίαση αλλά και αυτών που χρησιμοποιούν σε περιβάλλοντα Big Data και πως αποδίδουν οι υβριδικοί συνδυασμοί που χρησιμοποιούν αλγόριθμους επερωτήσεων κορυφογραμμής που δεν βασίζονται σε ευρετήρια στο ευρετηριασμένο σύνολο δεδομένων που δημιουργεί το SpatialHadoop.

Τέλος ένα από τα μεγαλύτερα προβλήματα που υπάρχουν κατά την διάρκεια ανάπτυξης ενός μοντέλου μηχανικής μάθησης είναι η ελλείψει επισημασμένων δεδομένων. Η έλλειψη αυτή γίνεται ακόμα πιο αισθητή σε περιβάλλοντα Big Data καθώς εκεί λόγω όγκου είναι πιο δύσκολη η επισήμανση τους. Στην βιβλιογραφία υπάρχουν πολλοί μηχανισμοί επισήμανσης δεδομένων ανάλογα με την εφαρμογή τους αλλά δεν υπάρχουν όμως μηχανισμοί για την αποδοτική επισήμανση μεγάλου όγκου δεδομένων. Στην πέμπτη συνεισφορά δημιουργήθηκε ένας μηχανισμός επισήμανσης δεδομένων που βασίζεται στο ερωτήματα κορυφογραμμής. Η χρήση ερωτημάτων κορυφογραμμής επιτρέπει την δημιουργία των ορίων αποφάσεως αποτελούμενων από μικρό αριθμό σημείων.

Λέξεις Κλειδιά: Ερωτήματα Κορυφογραμμής, Βελτιστοποίηση, χρονικά Ερωτήματα Κορυφογραμμής, Ανάστροφα Ερωτήματα Κορυφογραμμής, Γεωγραφικά Πληροφοριακά Συστήματα, Θαλάσσια Δεδομένα-Τεχνολογίες-Εφαρμογές, MapReduce, SpatialHadoop, Μεγάλα Δεδομένα, Ταξινόμηση, Όρια Αποφάσεων.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor Professor Manolis Maragoudakis for his support, encouragement and guidance during this study. I would also like to thank Dr. Theodoros Tzouramanis for his guidance and support on my early stages of my Phd studies.

A special thanks goes to my wife Eirini for being always next to me, in good times and bad.

I want to sincerely thank every person that helped and contributed on this long journey.

Finally, I would like to thanks my parents Vasilis, Eleni and my brother Alexandros for the moral and financial support they offered me during my studies as long as the love for sciences and gaining knowledge.

Dedicated to my family

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Living in the Information Age allows almost everyone to have access to a large amount of information and options to choose from to fulfill their needs. In many cases, the amount of information available and the rate of change may hide the optimal and truly desired solution. This reveals the need of a mechanism that will highlight the best options to choose among every possible scenario. Based on this the skyline query, which can be considered as a multi-objective optimization approach in database systems, was proposed. This decision support mechanism is based on Pareto optimality and retrieves the best options of a dataset by identifying the objects that present the optimal combination of the characteristics of the dataset. In this PhD Thesis we reason about data, big data management and supervised learning, which all of them can be part of decision support system.

1.1. Identifying Optimal Solutions

The rapid growth of decision support systems and the increasing size of multidimensional data lead researchers to seek for new efficient methods for data processing to retrieve useful insights. In many cases solving problems that require to identify the optimal solutions among multiple contradicting criteria is a difficult task since there may not exist a single optimal solution, or the computation of all the possible outcomes may be inefficient. Among the various approaches in multi-objective optimization some of these analytical methods may be rank-aware approaches that contain scoring functions. However, in many cases it may not be desired to define a cumulative scoring function to retrieve the best results of a dataset, since this will reduce the potential multi-dimensional comparisons of data to a single scalar value.

Taking this into account, Pareto optimality deflects from the strict ranking approach imposed by the rank-aware approaches that contain scoring functions and is directed to an approach that is more understandable by humans. This is different for example to top-k queries in database systems, where specific ranking functions and criteria are used, skyline queries assume that every user has a series of preferences over the attributes of data. Those preferences indicate what user's likes and dislikes (e.g. "I like the sea more than the mountains" or "I prefer to go vacations on an island rather than on a mountain). All the preferences are considered equivalent and this will help to discard the items of the dataset that will not be preferred by anyone. This results in a small subset that contains the most interesting and preferred items based on all the preferences of all users. This set will be the skyline set or an equivalent to the Pareto optimal set.

In recent years, skyline query processing has become an important issue in database research for extracting interesting objects from multi-dimensional datasets. The skyline query processing is applicable in many applications that require multi-criteria decision making without using cumulative functions to define the best results but based on user's preferences. The skyline operator filters out a set of interesting points based on a set of evaluation criteria from a potentially large dataset of points. A point is considered interesting, if there is not any other point better than that in all the evaluation criteria. The popularity of the skyline operator is mainly due the paradigm's simplicity and its applicability on multi-criterion decision support with respect to user preferences.

1.2. Optimization Approaches

In many cases the optimization problems can be classified in two categories named single and multi-objective optimization problems. The earliest approaches that tried to solve multi-objective problems tried to transfer them into single objective problems and then solve them with the classic approaches. In this case there was the need to define the degree of each objective function. One case could be the use of a linear function like the weighted sum method of all the objective functions. A categorization of Multi-objective trade-off optimization methods can be as in the following **Figure 1** where the Multi-objective optimization methods can be categorized to apriori, interactive and posteriori. In the priori approach the decision-maker defines his/her preferences in advance while in posteriori he/she identifies a set of optimal solutions to choose from. The interactive approach allows to the decision-maker to interactively identify the desired solution. In the interactive approaches there are solutions that provide exact and approximate solutions. In the approximate solutions there are methods that are based on the dominance like the Pareto

optimality or the skyline set of points, where the methods identifies a set of candidate solution/points. Among them he/she choose the one that maximizes his desired preferences. One of the most common algorithms to identify the set of Pareto points is the NSGA-II [1] in the scope of Pareto optimality and the BBS [2, 3] algorithm in the skyline queries.

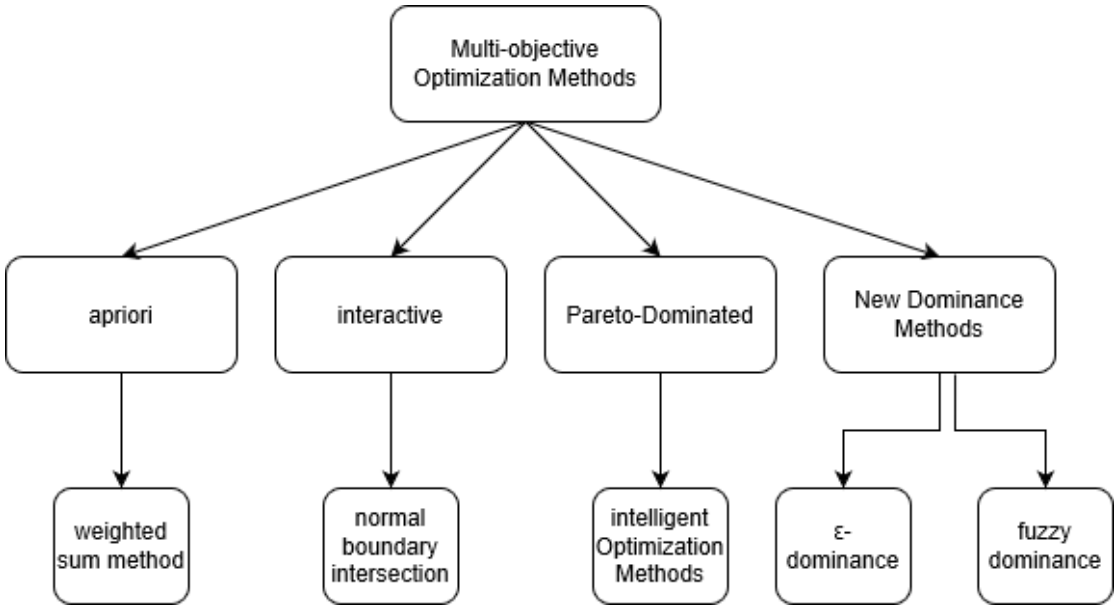


FIGURE 1: Optimization Approaches

1.3. A Multi-Objective Optimization Example

An example of a multi-objective optimization problem could be the case were and individual would like to choose the best job **Figure 2** that he would like to apply based on certain preferences. In the following example, the top 10 jobs are presented along with a median base salary for each job, a satisfaction factor and the number of job openings for each one of them. If an individual would like to select his dream job, he could do it in numerous ways by considering single or multiple criteria. Based on a single optimization criterion he could simply sort the jobs based on the satisfaction factor or the number of job openings. With that in mind he could select the Strategy Manager that has the greatest satisfaction factor or the Software Engineer which has the greatest number of job openings. If an applicant would like to select a job based on more than one criterion the problem becomes a multi-objective optimization problem and thus a little more complicated. Note that in this case there might not be a single optimal solution but rather a set of optimal solutions.

Furthermore, the optimization criteria might be contradicting, like the case of the base salary and the job satisfaction, where in some cases jobs with high salary have a low satisfaction factor. In this case an optimal combination would indicate the *Strategy Manager* if our optimization criteria are based on maximizing the salary and job satisfaction and the *Strategy Manager, Business Development Manager, Java Developer, Speech Language Pathologist and Software Engineer,* if the optimization criteria are to maximize the job satisfaction and job openings. For example, the *Strategy Manager* has the largest salary and satisfaction factor combined. On the other hand, a *Software Engineer* might be a good solution in comparison to the *Strategy Manager* since, despite the lowest satisfaction factor it has the most job openings.

Rank	Job Title	Median Base Salary	Job Satisfaction	Job Openings
1	Front End Engineer	$105,240	3.9	13,122
2	Java Developer	$83,589	3.9	16,136
3	Data Scientist	$107,801	4.0	6,542
4	Product Manager	$117,713	3.8	12,173
5	Devops Engineer	$107,310	3.9	6,603
6	Data Engineer	$102,472	3.9	6,941
7	Software Engineer	$105,563	3.6	50,438
8	Speech Language Pathologist	$71,867	3.8	29,167
9	Strategy Manager	$133,067	4.3	3,515
10	Business Development Manager	$78,480	4.0	6,560

Source: Glassdoor Economic Research (Glassdoor.com/research)

FIGURE 2: Top 10 Best Jobs in America in 2020

1.4. The Case of Skyline Queries

The computation of the skyline in database research is equivalent to determining the maximal vector problem in computational geometry [4], or equivalently the Pareto optimal set [4, 5] problem in operations research. The maximal vector problem is to find the subset of a set of vectors such that each one of them is not dominated by any other vector from that set. Considering that those vectors are points in a k-dimensional space, then the maximal vectors [6] can also be called admissible points [7] and the maximal set of vectors as Pareto set. This class of problems was extensively studied by the mathematical community in the 1960s.

As mentioned in [8] the skyline problem considers that the dataset cannot fit completely in the main memory (RAM) in order to be processed. This is more likely to be the case in modern database systems, where the dataset is retrieved from an external memory such as disks. Methods that do not rely on external memory are DD&C [4], LD&C [6] and FLET [9].

Authors in [10] proved that the initial algorithms, proposed for maximals [4, 5], which are based on the divide-and-conquer approach [11] (that divides the initial problem in equal sub-problems and then tries to solve each one separately, combining the results in the last step of the process) have quite bad performance with respect to the dimensionality of the initial problem. Additionally, these algorithms assume that the whole dataset fits into memory and they do not account for memory limitations and thus cannot be directly applied in a database scenario. Such kind of approaches suffers for the “curse of dimensionality” [12] which was first used by Bellman [13] and is often used to indicate that high dimensionality causes problems in resolving due to increased computational cost. This problem was observed and solved with the introduction of the skyline operator [8], which proposes a divide and conquer algorithm suitable for external memory and shows how it can be integrated into a database system.

Through the years many algorithms were proposed to efficient compute the skyline query problem either by using and indexed based approach or without using any index. Such an algorithm is he BBS [2, 3] which is an index-based algorithm that used the R-tree and a Branch and Bound approach. A non-index-based approach is the BNL [8] algorithm, which retrieves the whole dataset to identify final answer. Additionally, a large number of variations of the original skyline algorithm were proposed with the most common to be the Reverse [14] Skyline queries. Some indicative applications areas for which skyline queries [8] are useful are customer information services, decision support and decision-making systems. For instance, a skyline query can be used by travel agencies to find a reasonable priced hotel near the sea or to find good salespersons, which have low salary [8]. Additionally, reverse skyline queries [14] can assist in market research applications to find if a specific product is appealing to consumers or to identify the best location for a new branch store. Also it can be applied in economics [15], where it can

support microeconomic data mining or even in continuous data stream environments **[16]** such as stock exchange systems. Additionally, it can be used on location-based systems (LBS) in order to identify the shortest route to a destination or the closest point of interest among many **[17, 18]**. Another application is distributed query optimization. This can be particularly useful in cloud architectures, where data are scattered among servers or in the case where Quality of (web) services **[19]** is the primary goal. Skyline queries can also be used to focus on a subspace of attributes **[20]** in order to identify the skyline on a small subset of the dimensions of the dataset that are defined. Skyline queries have also applications in computer security and especially on problems concerning privacy **[21]** and authentication **[22]**. Skyline computation in metric space **[23]** can assist the *DNA searching problem* in bioinformatics. Finally, skyline queries are applicable in a wide variety of data types such as partial ordered **[24]** and incomplete **[25]** or uncertain data **[26, 27]**.

Many similar problems and operators related with skyline queries have been studied in the literature. For example, the Top-*K* query **[28]** retrieves the best *K* objects that minimize a specific preference function. The difference from skyline query is that the output changes according to a user-specified input function and the retrieved points are not necessary part of the skyline. The k-nearest neighbor (k-NN) query **[29]**, in another example, requires the existence of a query point *p* and outputs the k objects closest to *p,* in increasing order of their distance. In this case the difference from the skyline query is that k-NN query retrieves answers according to the proximity of a given point and not based on domination to other points. Finally, convex hull **[30, 4]** contains the points that are enclosed by the polygon that is defined from the minimum and maximum skyline (i.e. minimizing and maximizing values based on the evaluation criteria) of the given set of points. The main difference from a skyline is that it defines an area of interest rather than a line with individual interesting points.

1.5. Contributions

The main contribution of this Thesis is focused on four different topics. The first topic is data-related and analyzes the various data sources and requirements that a maritime information system has. Following a new query method that considers the temporal properties of the dataset is examined. Furthermore, a study was conducted to research new approaches in handling data over big data environments and especially in SpatialHadoop **[31]**. Finally, a new approach was studied on how to efficient estimate the decision boundaries in a classification process. More specifically,

the study on the data sources and requirements of a maritime information system identifies the type of data needed to implement such a system and surveys all the available data sources related to them. In addition, examines the restrictions in processing and distributing those data and gives useful insights about the real-life current needs in data processing. The key contributions of this study are:

- Defines the classes of data, which are valuable resources towards the development, performance tuning and efficient operation of maritime information systems
- Surveys both the open and restricted data sources that provide free-of-charge real-world geospatial data.
- Outlined data sources in international scale and special cases of sources that are significant for their propensity to provide specialized high-quality data relating to specific areas of the planet, such as specific countries or continents.
- Provide a thesaurus of high-precision real-word geospatial data to serve the needs of scientific research and development or educational work in the maritime information systems domain for purposes such as operational or benchmarking and experimentation or pattern recognition and data mining.
- Provides useful insights on the current needs in query processing, big data management and applied machine learning that will assist in identifying open research topics in relation to skyline queries.

The study on temporal skylines proposes the extension of the skyline query for temporal data and aims to demonstrate how the strategy for calculating the traditional skyline query is affected when also considering the time factor. Algorithms for processing modified versions of the static, dynamic, and reverse skyline queries for temporal data will be proposed. The key contributions of this study are:

- A new dominant method for evaluating temporal data using the skyline operator,
- Algorithms for computing temporal skylines and two of its well-known variants,
- An extensive experimentation on the efficiency of the above algorithms for optimizing the skyline query processing to handle temporal data.

The study of skyline and reverse skyline queries over SpatialHadoop proposes the extension of the skyline queries over Hadoop and especially the customized Hadoop implementations that intergrade non-distributed indexing methods like the r-tree. In summary, the key contributions of this study are the following:

- The proposal of an alternative approach to the one proposed in [32] for skyline query computation that will be used to enhance SpatialHadoop with reverse skyline queries.
- The proposal of a baseline algorithm for reverse skyline queries computation that incorporates a multiple filtering mechanism to allow for the pruning of the dataset as soon as possible.
- To perform experiments in large-scale synthetic, real datasets and different environments in order to demonstrate the performance benefits.

Finally, the study of a binominal skyline classifier proposes the use of skyline queries in estimating the decision boundaries in a classification problem. This approach best fits in Big Data environments, where the performance of well-known classifiers degrades due to the large number of data points. The key contributions of this study are:

- The decision boundaries are described by a small number of points even in a very large dataset; thus, a classification process needs to perform only a small number of computations to infer on the correct class.
- The decision boundaries can be independently computed, allowing for full parallelization of the whole modelling process.
- It is applicable in a wide range of multi-dimensional environments and specifically in any environment that its dataspace has an ordering, a feature that is inherited from the Skyline query family.
- The model can be easily explained and visualized allowing for greater interpretability.
- The decision boundaries can be easily transferred, reused and easily re-optimized allowing Transfer Learning.

Based on the work and results of this Thesis the following publications have been made:

Journal papers:

- Kalyvas, C., Kokkos, A., & Tzouramanis, T. (2017). A survey of official online sources of high-quality free-of-charge geospatial data for maritime geographic information systems applications. Information Systems, 65, 36-51.
- Kalyvas, C., & Maragoudakis, M. (2019). Skyline and reverse skyline query processing in SpatialHadoop. Data & Knowledge Engineering, 122, 55-80.
- Kalyvas, C., & Maragoudakis, M. (2020). A Skyline-based Decision Boundary Estimation Method for Binominal Classification in Big Data. Computation, 8.3:80.

Conference papers:

- Kalyvas, C., Tzouramanis, T., & Manolopoulos, Y. (2017, April). Processing skyline queries in temporal databases. In Proceedings of the Symposium on Applied Computing (pp. 893-899).
- Kalyvas, C., & Maragoudakis, M. (2020, September). A Skyline-based Decision Boundary Estimation Method for Binominal Classification in Big Data. In 2020 5th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM). IEEE.

Other:

- Kalyvas, C., & Tzouramanis, T. (2017). A survey of skyline query processing. arXiv preprint arXiv:1704.01788.

1.6. Thesis Structure

The next chapter will present the fundamental background on the skyline queries. At first will be presented the basic skyline algorithms that can be separated in indexed based and non-indexed based. The following chapter will present the variations of skyline queries that consist the skyline family and finally will be presented some of the applications of skyline queries.

The third chapter deals with the variety of the data and the various data sources. Through this study we understand the overall complex nature of the data that are available from the various open data sources and the need to build new efficient mechanisms to perform a numerous data related task. Additionally, we studied the percent of open sources available for a maritime-based information system and the various licenses applied.

The fourth chapter will deal with the temporal nature of skyline queries. Time is a fundamental part of our everyday life and can provide useful insights if studied along with existing queries. In this novel work we will present how time intervals can alter the final skyline result set.

In the fifth chapter we study the skyline queries in Big Data Environments. Since the volume of data needed to be analyzed continuously increases new mechanisms were designed such as Hadoop **[33]**. Furthermore, current indexing mechanisms like R-trees **[34]** were integrated in the Hadoop ecosystem. One of these approaches is the SpatialHadoop, which among other integrates a wide variety of classic index mechanisms.

The last chapter deals with efficient mechanisms to identify the decisions boundaries by exploiting the properties of skyline queries. The benefits for using this method is the small computation cost and the small number of points needed to describe the boundaries. Due to the small number of points produced they can also further be used to refine the boundaries without many additional computations.

2. LITERATURE REVIEW

In this section we will present the skyline problem and its properties, the fundamental skyline algorithms and their algorithmic approach in identifying the skyline set and the skyline family which consists of the variations of the initial skyline algorithm. To demonstrate the numerous algorithmic approaches for computing the skyline and the results produced by the various algorithms in the skyline family, an introductory example is presented that will be used in the rest of this section. This example considers how a typical skyline query is applied for a house purchase.

2.1. An Introductory Example

In this problem, it is supposed that a house might be of interest for someone if no other house is both cheaper and closer to a metro-station. It is considered that as the distance of a house from a point of highly (general) interest is decreased (in this case a metro-station), the objective value (price) of the house is increased. So, the user tries to find the best money-to-value ratio that satisfies his/her own preferences.

Table 1 presents a collection of eight houses that a user found to be sold in the vicinity of a particular metro station. Each row in the table contains information, which can be used to identify the most interesting houses. To make the example simple there exist only two numeric attributes (dimension) for the houses. One attribute will be price and the other will be distance from the metro-station. In this case first evaluation criterion is minimizing the distance from the metro-station and the second one is minimizing price. Every evaluation criterion is considered as a single dimension in the d-dimensional space.

House	price (in thousand €)	Distance from station (m)
H1	100	1500
H2	1400	500
H3	700	600
H4	1300	1000
H5	900	1300
H6	1600	100
H7	400	300
H8	200	1200
H9	1000	200
H10	500	1400
H11	500	900

TABLE 1: DATASET OF HOUSES

In **Figure 3** is presented the skyline of the existing set of houses. Houses H2, H3, H4, H5, H10 and H11 do not belong on the skyline as they are no one's top choice because for each one of them exists at least one house, which is better in terms of price or distance. Houses H1, H6, H7, H8 and H9 are the most interesting ones and so belong to the skyline. All the skyline points are connected by a line. The skyline is essentially the boundaries of the union of dominance area of all skyline point. To make it easily understood, the dominance area of a 2-dimensional point is the North-East quadrant of the space that occurs by imaginably drawing a x-y axis system with origin point the point of interest that is examined. The dominance area of a point will be inside the dominance area of a second point, noted as the first point dominates the second one, only if the first point is as good or better in all dimensions and better in at least one dimension based on the evaluation criteria. The skyline would refer to those points that are not dominated by any other point. In the house-metro station example "better" is minimizing the values.

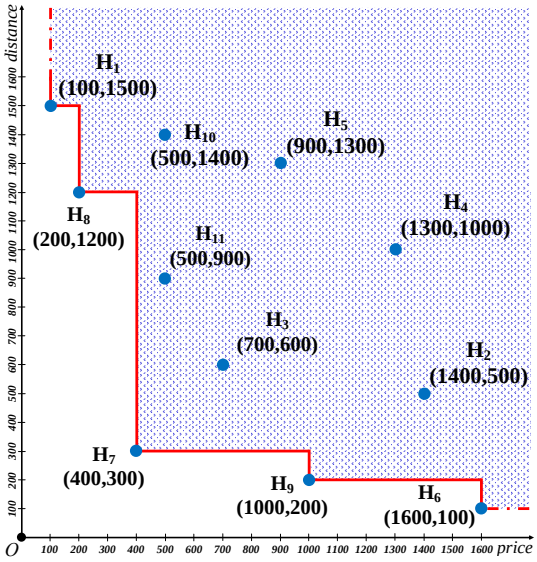


FIGURE 3: SKYLINE OF A SET OF HOUSES

Skyline queries can also involve more than two dimensions. For instance, a buyer could be interested in houses that are near to a metro-station, are cheap, have high square footage and low communal costs. The main idea of the skyline operator is to give the user the overall view of all interesting results and then let him/her to decide.

2.2. The Skyline Problem and its Properties

Skyline queries are a popular and powerful paradigm for incorporating user preferences into relational queries and extracting interesting points from a set of points. The main difference from the previous described problems is that instead of finding vectors or points, a skyline queries finds the maximals over a set of *tuples or the so-called* set of Pareto-optimal tuples. Those tuples are those that are not *dominated* by any other tuple in the same relation. One of the nice properties of the Skyline of a given set Ds of points is that any set of evaluation criteria that arise from user's preferences can be modeled in the form of a monotone scoring function $f: D_i \rightarrow R$, like L1 norm $f(x, y) = x + y$ or Euclidian norm $f(x, y) = \sqrt{x^2 + y^2}$. If p ∈ Ds and minimizes (or maximizes) the scoring function, then p is in the Skyline. That means, regardless how a user weights his/her preferences towards price and distance of houses, s/he will find a house that matches his/her preferences in the Skyline. In this example for simplicity, is assumed that skylines are computed with respect to minimum (min) conditions (minimizing the scoring function) on all dimensions. In particular, using the *min* condition, a point *p dominates* another point *r* if the coordinate of *p* on at least one axe is smaller than the corresponding coordinate of *r*, and no larger on any of the remaining axis. This implies that *p* is preferable to *r* according to any *preference* (*scoring*) function, which is monotone on all attributes. Furthermore, for every point **p** in the Skyline, there exists a monotone scoring function *f* such that **p** minimizes (or maximizes) that scoring function. This ensures that the skyline will contain all the preferable houses no matter how users weight their preferences. More formally, given a d-dimensional space D=*{d_1 ,…, d_d}* and a set *Ds* of points that belongs in D, a point p ∈ Ds can be represented as *P = {$p_{.d1}$,…,$p_{.dj}$}, 1<=j<=d,* where $p_{.dj}$ is the value of the *jth*-dimension of the point. Assume that the dataset Ds contains the points Ds={p_1 ,…, p_n}. The notation *pi.dj* ≥ 0*, with 1 ≤ j ≤ d and 1 ≤ I ≤ n,* is used to denote the *j*-th dimensional value of the *pi point.* Assume that for each dimension d_j there exists a total ordering relation, denoted by '<' or '>' according to the user's preferences. Without loss of generality in our examples we will use the '<' relation.

Definition 1: Dominate
Given points p, r ∈ Ds, p dominates r, denoted as p ≺ r, if and only if ∃ j∈ [1, d] such that p.dj<r.dj and ∀ i ∈ [1, d]-{j}: p.di ≤ r.di ■

Dominance has the property of a transitive relation. That is if p dominates r and r dominates t, then p also dominates t **(Figure 4)**. This is given more formally in the next proposition.

Proposition 1: Transitivity
Given points p, r, t∈ Ds, if p≺r and r≺t, then p≺t. ■

Transitivity can be used to eliminate from further consideration a single point or a group of points that are dominated by a point p, which in its turn is dominated by a new point r.

Through previous analysis, the domination between two points was explained. In contradiction if two points *p, r ∈ Ds* do not dominate (denoted with ≺≻) each other simultaneously (that is p≺≻r and r≺≻p) **(Figure 5)** are considered as incomparable in *Ds*, and denoted with *p ~Ds r or simply p ~ r.* More formally:

Proposition 2: Incomparability
Given two points p, r ∈ Ds, if p≺≻r and simultaneously r ≺≻ p, then p and r are incomparable on Ds (i.e. p~r). This property helps in determining if one or more points can be skyline points. A point in the skyline set must be incomparable to all other points of the set.

For example, consider the partitions on **Figure 5** that could be derived from the original dataset using a divide & conqueror approach. If we first examine partition 1 and identify at least a single skyline point (in any location inside of it) then partition 4 could be completely pruned. Furthermore, since partitions 3 and 2 are incomparable the skyline points in one partition do not affect the skyline points in the other partition and there is no case that points from partition 3 dominate points in partition 2 and vice versa.

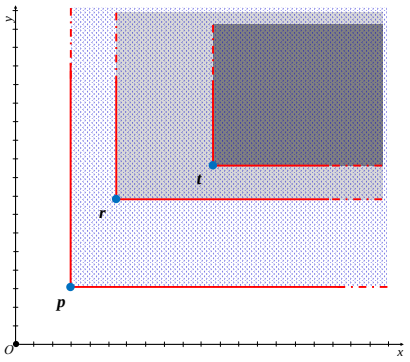


FIGURE 4: TRANSITIVITY DOMINANCE.

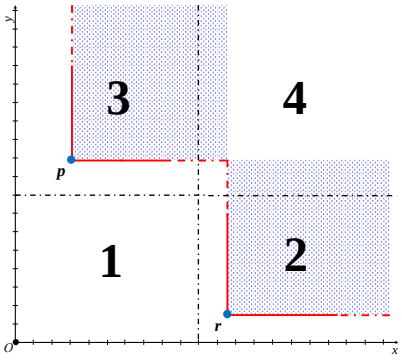


FIGURE 5: INCOMPARABLE POINTS.

The skyline of a dataset of n points refers to those points that are not dominated (are incomparable) by any other point. That is, a data point p is a skyline point if there does not exist any point on the dataset that dominates p.

Definition 2: Skyline
A data point p ∈ Ds is a skyline point iff ∄ r∈ Ds such that r≺p. ■

Notice that in order a point to be a skyline point it is not needed necessary to dominate another point in the dataset. Additionally, the skyline set of a dataset is unique.

The math notations that will be used in the subsequent discussion are summarized in **Table 2.**

Notation	Definition
D	d-dimensional space
Ds	Input dataset for skyline computation (set of points)
d	Number of dimensions of DS
d_i	one dimension ($1 \leq i \leq d$)
p, r, t	Data points
s	Skyline point
$p.d_i$	i-th dimension of the point p
q	query
S_{DS}	Set of skyline points of DS
f	Monotone function
$p \prec r$	p dominates r
$p \prec^q r$	p dominates r with regard to q
$p \prec_{Ds} r$	p dominates r in the Dataset Ds
$p \prec^q{}_{Ds} q$	p dominates r in the Dataset Ds with regard to q
$p \ ^{\varepsilon}\prec r$	p ε-dominates r
$p \prec\succ r$	p does not dominates r
$p \sim r$	p and r are incomparable
$S_{DS}{}^q$	Skyline set S of dataset Ds with regard to the query point q

TABLE 2: MATH NOTATIONS

Apart from the formal definition of the skyline there are some additional related interesting features. A skyline query tries to find an optimal solution for a user, based on *multiple, and sometimes conflicting, goals*. For example, a user may be interested in buying an economic house in Athens that is also close to a metro-station. In general case, houses that are near to a metro-station are expected to be more expensive (because they are preferred by the majority of buyers), therefore his/her preference for an economic house contradicts his preference for a house close to the metro station. Additionally, there may be *no single optimal answer* (or answer set) that satisfies exactly the preferences of the user, but rather there could be numerous answers that are close in satisfaction of his/her preferences. In the same example, it is unlikely that there exists a house that is the cheapest among all houses and is at the same location with the metro-station, (because houses near the metro-station are preferred by most buyers and a house in a distance will try to attract buyers with a lower price). Instead, one can expect to find in the skyline, among others, a list of economic houses such that those nearer to the metro-station to be slightly more expensive. Thus, users are typically looking for satisficing answers (decision making support). For the same query, different users with similar personal preferences, which are not exactly satisfied by a single optimal answer, may finally find different answers appealing. A person may be willing to pay a little more to be closer to the metro-station and another may be contented with a cheaper house as long as it is convenient to go by foot. In conclusion, it is important to present all interesting answers that may fulfill a user's need.

2.3. Fundamental Skyline Algorithms

Existing skyline computation methods can be classified into two categories, depending on whether (or not) to rely on pre-computed indexes on data. Index-based methods have better performance, since they avoid accessing the entire data collection, but have limited applicability due to the necessity of an indexed dataset. Additionally, multi-dimensional indexes like R-trees have their own limitations as they suffer from the well-known curse of dimensionality. Not index-based methods are more generic, in the sense that they do not require any specialized access structure to compute the skyline.

2.3.1. Block Nested Loop (BNL)

Authors in **[8]** introduces a Block Nested Loop (BNL) algorithm, which like the naive nested-loop algorithm repeatedly reads the set of tuples and eliminates points by finding other points in the dataset that dominate them. BNL allocates a buffer (window) in main memory that contains a number of points to sequentially track the dominance between them **(Figure 6)**. The algorithm reads the input data and each point is retrieved and compared against the points in the buffer. In the first run of the algorithm no point will exist in the buffer so it is trivial to insert the first point in the buffer. For the next runs if the point retrieved is dominated by at least one point in the buffer there is no need to continue the comparison with the others points that maybe exist in it and the point is discarded. Otherwise if the point is incomparable or dominates one or more points in the

buffer, those points that are dominated are removed from the buffer and the new point is inserted. **Figure 6** illustrates the algorithm in its fifth iteration in which has processed houses H1, H2, H3, H4, H5 from the input dataset. As seen points H4 and H5 are dominated by one or more points in the buffer and so are discarded from further processing. For further considerations suppose that the buffer has size 3, meaning that can store up to three entries.

If in any stage the buffer becomes full, a different approach is followed. Once this happens, the rest of the input is processed differently and a temporary overflow disk file is used **(Figure 7)** to store the points that were compared and characterized as incomparable or dominated existing points in the list and cannot be further placed in the window. Such a point is house H6, which is incomparable with the houses already existing in the buffer and thus is placed on the temporary file. Nevertheless, the dominated points in the window are still discarded as before right after each dominance comparison. After the dataset has been read now the temporary file is used as input for the next passes of the algorithm. After the first run all the points of the input will be either inside the window or in the temporary file. Points that inserted in the window before any other point was inserted in the temporary file are guaranteed to be skyline points. This can be checked by assigning a timestamp to each point that exists in the window and the temporary file.

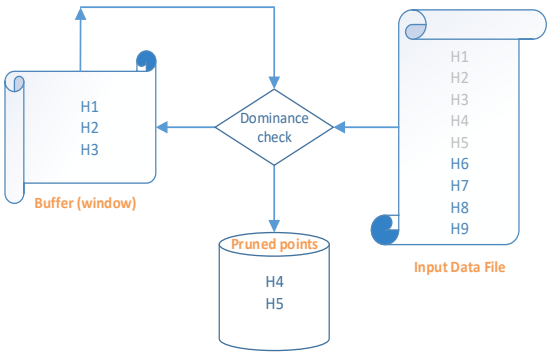


FIGURE 6: BNL WITHOUT TEMPORARY FILE.

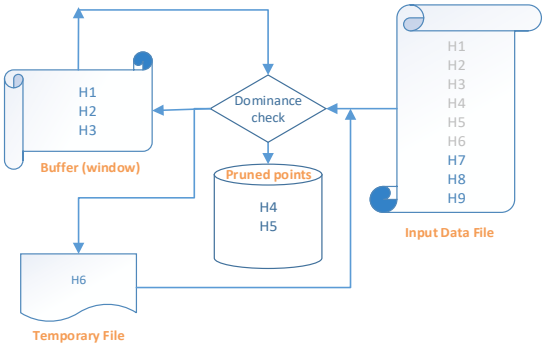


FIGURE 7: BNL WITH TEMPORARY FILE.

The algorithm may require a large number of passes until the complete skyline is computed and eventually terminate as at the end of each pass the size of the temporary file will be decreased. BNL works well if the size of the resulted skyline is small and in best case fits into the window, which will result in the termination of the algorithm in one iteration. BNL algorithm cannot compute skyline points progressively. Its performance is very sensitive to the number of dimensions and to the underlying data distribution. Especially, it is good for up to five dimensions for a uniform distribution, but its performance degrades if the distribution tends towards an anti-correlated distribution.

2.3.2. Divide & Conqueror (D&C)

The divide-and-conqueror (D&C) algorithm proposed in **[8]** is an extension of the two-way partitioning divide-and-conqueror algorithms proposed in **[4, 5].** These earlier proposed algorithms do not scale well for large datasets, since they do not take into account main memory limitations. The D&C algorithm recursively divides the input dataset in m partitions $\{P_1,\ldots P_m\}$ (m-way partitioning), in order for each of them to fit in the main memory **Figure 8.** The partitions boundaries are determined by computing the q-quintiles of the dataset, which results in the division of the dataset into q-1 equal subsets. Then a local (partial) skyline S_i is computed for each partition P_i with $1 \leq i \leq m$. Finally, the algorithm computes the global skyline by progressively merging the local ones based on a bushy merge tree **Figure 9** and **Figure 10**. This way points that belong to one partition and are dominated by points of another partition can be removed. The points that left from the merging process are the skyline points and the algorithm terminates returning the resulted set.

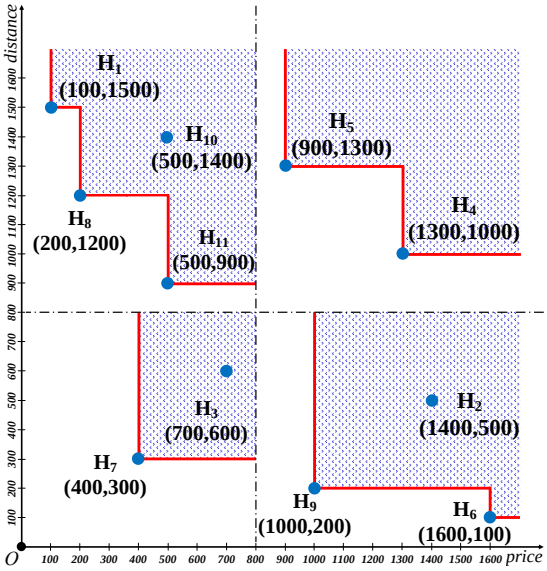


FIGURE 8: DIVIDE AND CONQUEROR

As BNL algorithm, so to the D&C cannot produce skyline points progressively since the first skyline point can be generated only when the entire dataset has been scanned. Moreover, as the main memory size increases it performs better as it requires the partitions to be in-memory. D&C is less sensitive than the BNL to the number of dimensions and correlations in the database.

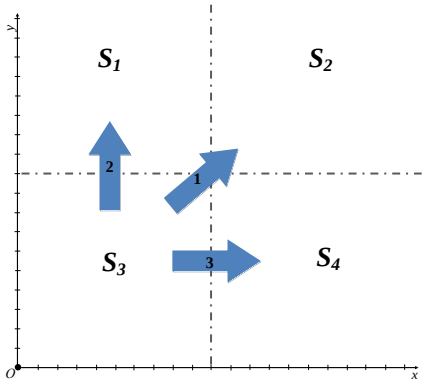


FIGURE 9: MERGING PROCESS

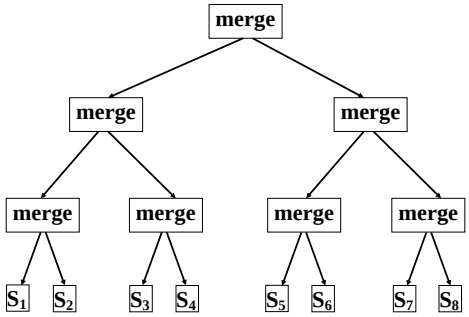


FIGURE 10: BUSHY MERGE TREE

2.3.3. Bitmap

To resolve the problem of progressive skyline computation, [35] proposed the index-based Bitmap algorithm, which encodes all data into a bitmap structure to identify the skyline points by exploiting the speed of a bitwise & operation. Bitmap is a progressive algorithm, which means that it does not need to scan the complete dataset to return results and is based on a bitmap structure, which encodes all the information required to determine if a point belongs in the skyline.

To describe the algorithm assume that a point p={$p_{.d1}$,…., $P_{.dj}$ }, 1≤j≤d in a d-dimensional space is represented by an m-bit vector. From those m-bits each p.di is represented by a number of ki bits. Each ki has as many bits as the number of distinct coordinate values of all the points of the dataset in that dimension and thus $m = \sum_{i=1}^{d} k_i$.

To incorporate the house-metro station example, there are 10 distinct values for dimension price and 11 distinct values for dimension distance. That means k1+k2=10+11=21 and thus m=21. Considering the *min* annotation and assuming that $p.d_m$ is the j-th smallest number on the i-th dimension it can be represented by the k_i bits setting the (k_i-j_i+1) most significant bits to 1 and the rest to 0. In detail, value 400 is the third largest value among the 10 in the first dimension so in its bit-representation the first (10-3) +1=8 most significant bits will be assigned to 1 and all the

other to 0. The results of the mapping process are shown in **Table 3.** Next the algorithm needs to determine if a point is a part of the skyline or not.

In our case we will check points H7, which from the previous example is a skyline point and H4, which is an ordinary point. For the algorithm to compare the points, it obtains the array of bit-vectors of all points and transposes it in an m-length array of bit-slices. Each bit-slice V_i corresponds to the sum of the i-th bit-value of the dimension, of all points. The bit-length of bit-slices depends on the number of points. The two bit-slices of House H7 for the two dimensions are shown in bold in the **Table 3**.

After the construction of the bit-slices the algorithm performs 3 bitwise operations among 2 sets of bit-slices. The first set contains the bit-slices V_x, V_y (one for each dimension), where the last bit of the point resides, which is equal with one. The second set contains the next in order bit-slices V_{x+1} , V_{y+1} of those that selected in the previous set. In the case that the bit-slices of the previous step is the last in order, then is used the zero bit-slice (all bits zero). The first bitwise operation A will be an AND operation between V_x and V_y. The second bitwise operation B will be an OR operation between V_{x+1} and V_{y+1} . The third bitwise operator C would also be an AND operation between the results of the two previous operations. If the result of the final operation is zero then the tested point is a skyline point.

For point H7 $A=V_x$ AND V_y ={10111011111 AND 00000110100} = 00000010000, which indicates that the points that have values in each dimensions that are greater or equal to this point is only the point H7. The second operations B= V_{x+1} OR V_{y+1}= {10000001000 OR 00000100100} = 10000101100, which shows that points which have some of it is dimension better than H7 are the points H1, H6, H8 and H9. The last operation C=A AND B={00000010000 AND 10000101100} = 00000000000 which shows that there is no house that dominates H7.

In the case of House H4 $A=V_x$ AND V_y ={10111011111 AND 01110110101}= 00110010101 which indicates that houses H3, H4, H7, H9, H11 are equal or better in each dimension. Operation V_{x+1} OR V_{y+1}= {10101011111 OR 01100110101} = 11101111111 indicates that points {H1-H3}, and {H5-H11} are better in at least one dimension from H4. The final operation C=A AND B = {00110010101 AND 11101111111} = 00100010101 indicates that points H3, H7, H9, H11 dominate point H4.

House	coordinates	Bitmap representation
H1	(100 , 1500)	(1111111**1**11 , 10000000**0**00)
H2	(1400 , 500)	(1100000**0**00 , 11111111**0**00)
H3	(700 , 600)	(1111110**0**00 , 11111110**0**00)
H4	(1300 , 1000)	(1110000**0**00 , 11110000**0**00)
H5	(900 , 1300)	(1111100**0**00 , 11100000**0**00)
H6	(1600 , 100)	(1000000**0**00 , 11111111**1**11)
H7	**(400 , 300)**	(1111111**1**00 , 11111111**1**00)
H8	(200 , 1200)	(1111111**1**10 , 11110000**0**00)
H9	(1000 ,200)	(1111000**0**00 , 11111111**1**10)
H10	(500 , 1400)	(1111111**0**00 , 11000000**0**00)
H11	(500 , 900)	(1111111**0**00 , 11111100**0**00)

TABLE 3: BITMAPPED DATASET

Even if Bitmap is a progressive algorithm it must consider all points of the dataset to compute the full Skyline, which tends to be an expensive operation because for each point inspected must retrieved the bitmaps of all points. Additionally, the algorithm does not allow the user to give preferences in which order the results are produced but rather points are returned depending on the clustering of the data. Finally, bitmaps perform well when the number of distinct values per dimension is small.

2.3.4. Index

Among the Bitmap algorithm **[35]** authors additionally proposed the *Index* algorithm, inspired from the rank aggregation algorithm proposed in **[36],** which partitions the entire d-dimensional dataset

into d ordered lists. It uses a specialized B-tree to index each point by a transformation mechanism that maps high-dimensional points into single dimension point. Note that it can use any single dimension index structure and not only a b-tree. The data points are mapped to y and ordered as $y=d_{min} + x_{min}$. A point $p = (p.d_1, p.d_2, \ldots, p.d_d)$ of the dataset belongs to the i-th list $(1 \leq i, j \leq d)$ if it's $p.d_i$ value is minimum among all p.dj values, that is $p.d_i \leq p.d_j$ for all $i \neq j$. In each list points are organized in batches and sorted in an ascending (or non-ascending) order of their distinct minimum (or maximum) value in that dimension. Each batch is identified by the minimum value of the point that represents. Points with the same minimum value in each list are organized in the same batch. Each batch is processed according to its ascending index value and the algorithm tries to determine if it belongs to the skyline. If a batch has more than one point a local skyline is computed, which is then checked if it can be merged to a global one.

In the case of the hotel metro-station example the houses that belong to the first list and have their first coordinate minimum among the two are H1, H5, H8, H10, H11 and houses that have their second coordinate minimum and belong to the second list are Houses H2, H3, H4, H6, H7 and H9 **(Table 4).** Houses H10 and H11 have the same minimum value so they belong to the same batch. When the algorithm starts, it loads the first batch from each list. The two first batches have minimum value 100, so the algorithm process with the batch from the first list. Point H1 is added to the skyline list because is a single point and the skyline list is empty. The next batch the algorithm handles is H6, which was considered previously. The point is incomparable so is added to the list. The next batches from each list loaded are H8 and H9 that again have the same Min value, which is 200, so the algorithm continues with the one on the first list. H8 is incomparable so it is added to the list. The next point in the first list has Min value 500 so the algorithm continues with the previous considered H9, which is added to the list. Algorithm continues by loading the batches {H10, H11} and H7 from each list respectively. The batch with the smallest minimum value is H7 which is added to the list because it's not dominated by any point in it. At this step the algorithm terminates because both the coordinates of H7 are smaller than or equal to the minimum value of the next batch {H11, H10}, H2 on the two lists. In this case the algorithm does not need to proceed further because all the remaining point will be dominated by H7 and thus algorithm terminates returning the set of skyline point.

For clarity, we explain what should happen in the case {H11, H10} was processed. This batch has two points so in that case the algorithm would calculate the local skyline of the batch. The resulted point (or points) would be checked if they could be a part of the skyline. Both points do not dominate each other so both belong to the local skyline. In this case algorithm will check both points if can be added to the skyline list.

Min_1	**Dimension 1**	**Dimension 2**	**Min_2**
Min_1= 100	H1 (100 , 1500)	H6 (1600 , 100)	Min_2= 100
Min_1= 200	H8 (200 , 1200)	H9 (1000 , 200)	Min_2= 200
Min_1= 500	{ H11 (500 , 900) , H10 (500 , 1400) }	H7 (400 , 300)	Min_2= 300
Min_1= 900	H5 (900 , 1300)	H2 (1400 , 500)	Min_2= 500
		H3 (700 , 600)	Min_2= 600
		H4 (1300 , 1000)	Min_2= 1000

TABLE 4: INDEX APPROACH

The Index algorithm can quickly return skyline points in bursts (since it examines collection of points together) but does not support user-defined preferences since the order of the skyline points that are returned is fixed and depends on the value distribution of the data.

2.3.5. Nearest Neighbor (NN)

The Nearest Neighbor (NN) algorithm **[37]** is the first algorithm that uses the widespreaded R*-tree **[38, 34]** index structure to massively eliminate points by avoiding redundant dominance checks. The algorithm recursively applies the NN search, using an existing algorithm such as **[39, 40]** which is based on any monotone distance function (i.e. L_1-norm or Euclidean norm(L_2-norm)). At the beginning an NN search is applied to find the point with the minimum distance (mindist) from the beginning of the axes (when the problem is to be minimized) and inserts the resulted point into the skyline. This point partitions the space in four partitions. One partition contains only

points that are dominated by this point and thus can be removed. A second partition contains no point according to NN search and the other two partitions will be processed recursively through a to-do list to output the skyline result. If a region is empty is not sub divided any further and is removed from the to-do list. The algorithm terminates when the to-do list is empty.

Algorithm starts by searching for the nearest neighbor from the origin point defined (in this case the start of axes). The nearest point to the origin is H7 (400,300) with mindist 700, based on the L_1, and is guaranteed to be a skyline point. This point partitions the dataspace into four regions as presented in **Figure 11**. Region 1 contains no points according to the definition and properties of nearest neighbor. Region 4 contains all the points that have greater coordinate values than those of the nearest neighbor point. Thus, the points that belong to this region are dominated by the NN point and so they can be pruned massively (this could efficiently done with the r-tree implementation). Region 2 contains the points [0, 400) [300, ∞) and region 3 that contains all the points that belong to [400, ∞) [0, 300). The set of partitions resulted after the discovery of a skyline point must be inserted in a *to-do* list so the algorithm removes the initial region and inserts in their position regions 2 and 3, which are needed to be investigated. The algorithm recursively calls itself on Region 2 and Region 3.

In the recursion of Region 2, which is the first region of the to-do list, the algorithm makes again an NN search to find the next skyline point. The NN point that is retrieved on R2 {[0,*400*) [300, ∞)} is H8 (200, 1200) with mindist 1400 which is inserted in the skyline list. Due to the discovery of the NN point Region 2 is divided in 4 partitions **(Figure 12)**. Region 2.1 will be [0,200) [300, 1200), region 2.2 [0,200)[1200, ∞), region 2.3 [200,400) [300, 1200), and region 2.4 [200,400)[1200,∞). As mentioned before regions 2.1 and 2.4 are not needed to be considered. As a final step the algorithm removes region 2 from the to-do list and inserts regions 2.2 and 2.3. Next the algorithm will recursively call itself on the first region on the to-do list which is region 2.2. An NN query in this region will return point H1 (100, 1500) with mindist 1600 that is added to the skyline list. Due to the discovery of the NN point, region 2.3 is divided in 4 partitions which are region 2.2.1 [0,100)[1200,1500), region 2.2.2 [0,100)[1500,∞), region 2.2.3 [100,200)[1200,1500) and region 2.2.4 [100,200)[1500,∞). As previously regions 2.2.2 and 2.2.3 are inserted in the To-Do list and processed recursively.

The algorithm will perform the next NN query starting with the first region on the to-do list which is 2.2.2. The region is empty, so the algorithm discards it and process the next one. Region 2.2.2 is also empty so it is discarded and so the region 2.3. The only region remaining is Region 3. As with Region 2 the algorithm will recursively call itself until the to-do list is empty, where in that case algorithm terminates and returns the final skyline list. The skyline points returned by processing Region 3 are H9 (1000, 200) with mindist 1200 and H6 (1600, 100) with mindist 1700.

The algorithm improves the divide-and-conquer algorithm by applying the D&C framework on datasets indexed by R*-trees. NN is the first algorithm that gives the user control over the process by tendentiously selecting on-demand the preferred region to be processed and allows him/her to give preferences by altering the scoring function on-the-fly. On the downside, the algorithm has large I/O overhead, especially in high dimensional spaces, due to the recurrent access of the R*-tree. Additionally, the to-do list size may exceed the size of the dataset for as low as 3 dimensions **[2]**. Finally, is mentioned that in the general case of $d>2$, regions overlap in such a way that the same Skyline point can be found more than once. For that reason, authors proposed some additional elimination methods for datasets with d>2.

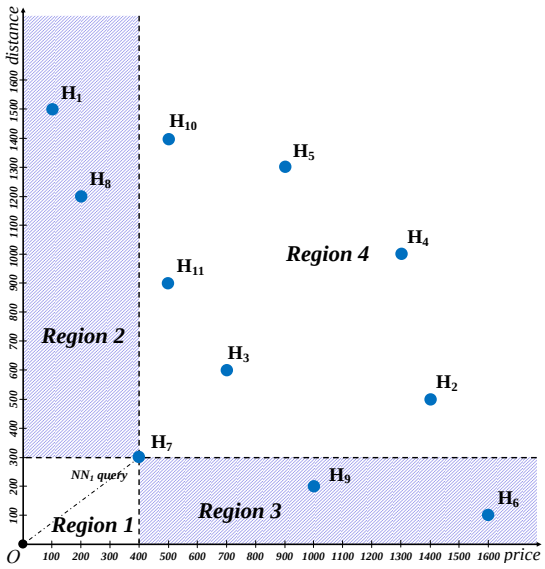


FIGURE 11: REGIONS AFTER 1ST NN QUERY.

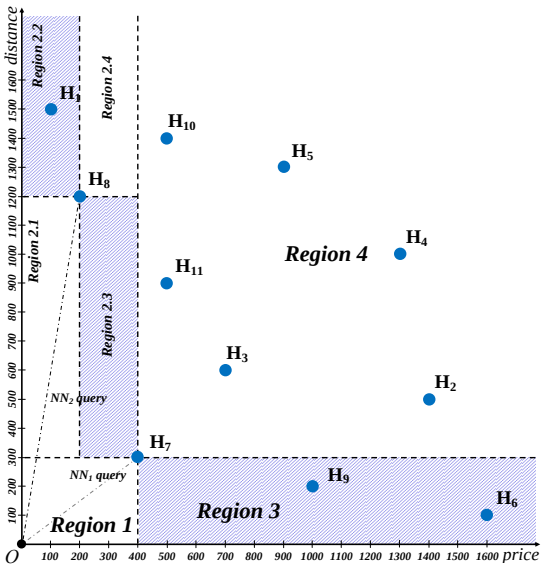


FIGURE 12: REGIONS AFTER 2ND NN QUERY.

# of NN query	To-do List space partitions	Skyline Points
0	[0,∞),[0,∞)	∅
1st	R2{ [0,*400*) [300, ∞) } and R3{ [400,∞) [0,*300*) }	H7
2nd	R2.2{ [0,200)[1200, ∞) }, R2.3{ [200,400)[300,1200) } and R3{ [400,∞) [0,*300*) }	H7, H8
3rd	R2.2.2{ [0,100)[1500,∞) } , R2.2.3{ [100,200)[1500,∞) } , R2.3{ [200,400)[300,1200) } and R3{ [400,∞) [0,*300*) }	H7, H8, H1
4th	~~R2.2.2{ [0,100)[1500,∞) } , R2.2.3{ [100,200)[1500,∞) } , R2.3{ [200,400)[300,1200) }~~ and R3.2{ [400,1000)[200,300) } , R3.3{ [1000,∞)[0,200) }	H7, H8, H1, H9
5th	~~R3.2{ [400,1000)[200,300) } ,~~ R3.3.2 { [100,200)[100,1600) } , R3.3.3{ [1600,∞)[0,100) }	H7, H8, H1, H9,H6
6th	Empty	H7, H8, H1, H9,H6

TABLE 5: TO-DO LIST BASED ON NN QUERY.

2.3.6. Branch and Bound Skyline (BBS)

Both NN and Branch and Bound Skyline algorithm (BBS) [2, 3] apply nearest neighbor search techniques mentioned previously to progressively output skyline points from datasets that are indexed by *R**-trees to massively eliminate points from being checked for dominance. BBS algorithm is an improvement of NN algorithm. In contradiction with NN that searches R*-tree many times, BBS traverses the R*-tree once. Table 6 illustrates the indexed dataset. Data points are organized in the R*-tree, in which each internal R-tree node can hold up to three entries, and that each leaf node can hold also up to three entries. In the example, an intermediate entry e_i of the R-tree of Figure 13 corresponds to the minimum bounding rectangle (MBR) of a node N_i of the R-tree, while a leaf entry corresponds to a data point H_i (Figure 14). As in the NN algorithm mindist denotes the minimum distance of a point or an MBR from an origin point. The *mindist* of a point is computed according to the L1 norm as the sum of its coordinates and the *mindist* of a MBR as the *distance* of its lower-left corner from the origin point. The algorithm uses the best-first search paradigm to traverse the R-tree, in such order that it always evaluates and expands, among all un-visited nodes, the tree node closest to the origin. All the candidate entries are kept in a *heap* until they are no longer useful. Entries in the heap are sorted in ascending order of their mindist. Skyline points are generated iteratively and stored in a list *in* the main memory, for dominance validation. Initially, the root of the *R*-tree is inserted in the heap. At each step, the top heap entry with the smaller mindist is removed. If it is a R*-tree node, its children, which are not dominated by any current skyline point, are inserted into the heap. If it is a point (leaf node), it is tested for dominance with the skyline points found so far by issuing an enclosure query. If the examined point (or region) is entirely enclosed by any skyline candidate's dominance region, then the point (or the entire region) is dominated. Notice that every entry is checked twice for

dominance because an entry in the heap may become dominated by skyline points discovered after its insertion. In the end all the points, except of those where one of its ancestor nodes has been pruned, will be examined. To efficiently examine the dominance relationship, is maintained an in-memory *R*-tree that contains the skyline points found so far. When the heap is empty the algorithm terminates. Initially, BBS inserts all the child entries of the root of the *R*-tree into the heap.

The algorithm begins with region e_1 in its heap. As it proceeds it iteratively processes the (leaf/intermediate) entry which has the minimum *mindist* value and if it's an intermediate entry e_i is expanded and its non-dominated children are inserted to the heap, ordered by their *mindist*. After the expansion of e_3 the first entry of the heap is a leaf node. The list of skyline points is empty so H_7 is inserted in the list. Next e_8 is expanded and H_9 is inserted to the list since it is not dominated by H_7. *In the next* step e_2 is expanded. Region e_4 is inserted in the heap, but region e_5 is dominated by the found skyline point H7 so the region is discarded. Next region to be expanded is e_4 . The points on the heap are sequentially checked if they are dominated by any so-far found skyline point and if not are inserted to the list. From this comparison points H_8, H_1, H_6 are inserted to the list and point H_8 is discarded. Now the only region left in the heap is e7 which is not expanded because is dominated by the skyline point H_7 *and* H_9.

Action	Heap contents	Skyline points
Initial state	(e_1,200)	∅
Expand e_1	(e_3,500), (e_2,1300)	∅
Expand e_3	(e_6,700), (e_8,1100), (e_2,1300), (e_7,1800)	∅
Expand e_6	~~**(H_7,700)**~~, (e_8,1100), (e_2,1300), (H_3,1300), (H_{11},1400), (e_7,1800)	H_7
Expand e_8	~~**(H_9,1200)**~~, (e_2,1300), (H_3,1300), (H_{11},1400), (H_6,1700), (e_7,1800)	H_7, H_9
Expand e_2	(e_4,1300), (H_3,1300), (H_{11},1400), (H_6,1700), ~~(e_5,1800),~~ (e_7,1800)	H_7, H_9
Expand e_4	~~(H_3,1300), **(H_8,1400)**, (H_{11},1400), **(H_1,1600)**, **(H_6,1700)**,~~ (e_7,1800)	H_7, H_9, H_8 ,H_1 , H_6
~~Expand e_7~~	empty	H_7, H_9, H_8 ,H_1 , H_6

TABLE 6: HEAP CONTENTS OF BBS

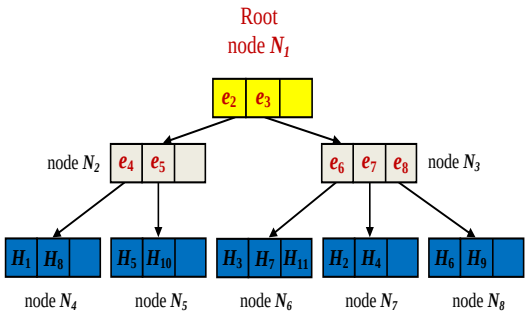


FIGURE 13: DATASET INDEXED BY THE R-TREE

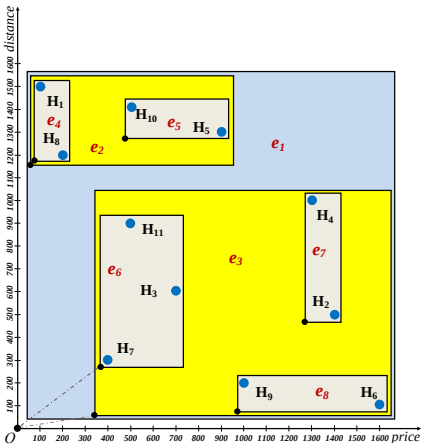


FIGURE 14: MINIMUM BOUNDING RECTANGLES (MBRS)

One of the most important properties of BBS is that it guarantees the minimum I/O costs and equivalently R-tree page accesses. Additionally, along with the NN algorithm can incorporate user preferences in general skyline computation. However its performance can deteriorate due to many unnecessary dominance checks and due to high dimensionality based on the curse of dimensionality [12] since an R-tree is efficient for up to 5 dimensions.

2.3.7. Sort Filter Skyline (SFS)

The sort-filter-skyline (SFS) algorithm [41] improves BNL performance by presorting the input dataset in an ascending order according to a monotone preference function f, such as the sum of coordinates of a point on all dimensions, or optimized as entropy (assuming in both cases that values have been normalized in (0,1) non-inclusive). Presorting enforce that a point *p* dominating

another point *q* will be visited before *q*. This ensures the progressive behavior of SFS and the reduction of the number of pairwise comparisons between points. The algorithm examines the data points by the ascending order of their scores and keeps an in-memory buffer that has the till now found skyline candidate points in a similar way as that on BNL. At beginning the buffer is initially empty. A point is read from the sorted dataset and if it is not dominated by a skyline point in the buffer is inserted into it. The dominance tests in SFS are performed by an exhaustive search on the existing skyline points.

Authors have found that the entropy scoring function $E_D(p) = \sum_{i=1}^{d} \ln(p'.d_i + 1)$, where p'.$d_i$ is the normalize value of *p.di* in (0,1) non-inclusive, yields the most effective discarding during the skyline computation. Intuitively, the smaller entropy value a point has, the less likely is to be dominated.

Value Normalization: There are several ways to normalize the values of a dataset. One case is to divide all the values of the dataset with the maximum value found over this. That is $f(p.d_i) = (p.d_i/max)$, where max is the maximum value observed in the dataset. In this case the dataset would be normalized in the [0,1] inclusive which uses efficiently all the range of [0,1]. But in this example, this is not the case because the values are needed to be normalized in (0, 1) non-inclusive. A case to achieve this is by dividing all the values of the dataset with a higher value than the maximum value of the dataset. For memorization reasons and simplicity of numbers and computations we choose to divide all values by 10.000. That is $f(p.d_i) = (p.d_i/10000)$. We note that this is not considered as a user preference since in another case we could assume that the distance between the two furthest locations of the town is 2000 meters and divide with this number.

To demonstrate the algorithm the entropy scoring function was used. Points of the dataset of the house metro-station example will be normalized and sorted as in **Table 7** by an ascending order of their score and will be processed in this order. The first point that is inserted in the buffer is H7 since the buffer is empty. The second point is H9 which is incomparable to H7 so its inserted to the buffer. H3 is dominated by H7 so it is discarded. H8 is incomparable with H7 and H9 that are already in the buffer so it is added. Point H11 is dominated by H7 so it is discarded. Point H1 is incomparable to the points that belong in the buffer so it is added to the buffer. The same hold for point H6. The rest four points are dominated by a point in the buffer (H2 by H7 and H9, H10 by H7 and H8, H5 by H7 and H8, H4 by H7 and H9) so they are discarded. It is observed that the killer-dominant points are first in the presorted dataset which ensures maximum discarding with minimum comparisons. An indication for this is that the algorithm processed seven points to find the five skyline points out of the total eleven points.

House (h_i)	Price	Distance	$E_D(h_i)$	# points that dominate
H7	0,04	0,03	0,068779515	6
H9	0,1	0,02	0,115112807	2
H3	0,07	0,06	0,125927557	-
H8	0,02	0,12	0,133131313	2
H11	0,05	0,09	0,13496786	-
H1	0,01	0,15	0,149712273	0
H6	0,16	0,01	0,158370336	0
H2	0,14	0,05	0,179818427	-
H10	0,05	0,14	0,179818427	-
H5	0,09	0,13	0,208395329	-
H4	0,13	0,1	0,217527813	-

TABLE 7 : PRE-SORTED DATASET

The main drawback of SFS is that it cannot adapt to different user preferences and must scan the entire dataset to return a complete skyline, as with BNL. Nevertheless, it can be stopped early returning some of the skyline points. The significant advantage over BNL is that reduces the number of comparisons needed.

2.3.8. Linear Elimination Sort for Skyline (LESS)

Skyline algorithm Linear Elimination Sort for Skyline (LESS) **[10]** is an optimized version of SFS, which achieves a better average performance. As with SFS it sorts the dataset based on the

entropy scoring function which has the advantage of pushing the killer-dominant points in the beginning of the sorted dataset. The algorithm implements two optimizations.

The first optimization in the first pass of the external sorting process makes use of a buffer called elimination-filter (EF), which keeps a small set of points (copies of them) that have the best entropy scores seen so far. This set will be used to prune efficiently and as early as possible the dominated tuples of the dataset. The input dataset is divided in b blocks and each block of points is read in order to be sorted (i.e. using quicksort). During sorting the algorithm compares the points of the block with those of the EF. If the point from the block is dominated by a point in the EF, it is discarded. Otherwise if the point is incomparable or dominates other points in the EF it is inserted (a copy of it) in the EF and the points of EF that are dominated are discarded. It is noted that points of the EF buffer are not guaranteed to be maximals.

The second optimization combines the final pass of the external sorting process (last merge step of the b blocks) with the first pass of the skyline-filter (SF) process (i.e. first pass of the BNL component of SFS), which eliminates the remaining dominated tuples to get the final skyline. As in SFS and BNL, may be required multiple passes of the SF component to compute the final skyline. If the SF buffer becomes full, then an overflow file will be created. In general, the EF filter reduces effectively the size of the input dataset that will be processed by the SF process and additionally the combination of the final pass of the EF process with the first pass of the SF process saves always one pass from the computation of the skyline. LESS is not be applicable in scenarios in which one has no direct control on the algorithm used to sort tuples. Additionally, as in SFS all points on the dataset should be scanned at least once after sorting.

2.3.9. Sort and Limit Skyline Algorithm (SaLSa)

The Sort and Limit Skyline Algorithm (SaLSa) algorithm **[42]** is an improvement of SFS and LESS which strives to avoid scanning the complete sorted dataset as opposed with the two previous algorithms. As SFS and LESS, it does not have an index structure and is the first algorithm that exploits the values of a monotone scoring (limiting) function to sort the dataset and effectively limit the number of point to be read and compared by using a threshold value.

The author's suggestion is an optimal sorting function, which orders the points according to the value $fmin(p) = (\min_{i\in[1,d]} p.d_i, sum(p))$, which is the minimum coordinate value of a point among all dimensions and $sum(p) = \sum_{i=1}^{d} p.d_i$ is the second sorting element that works as a tie-breaking rule. Letting S be the current set of skyline points, for each point $p_i \in S$ let $\underline{p_i} = \max_j\{p_i.d_j\}$, which is the maximum coordinate value of a point. The threshold value that is used during the filter-scan process to check whether all points in the rest of the sorted dataset are dominated and for the algorithm to stop, is set as $p_{stop} = \arg\min_{i\in S}\{\underline{p_i}\}$. That is P_{stop} equals with the minimum $\underline{p_i}$ value calculated so far based on the existing skyline points. The computation of P_{stop} can be done incrementally by simply updating the value at each skyline point insertion in O(1) time.

The algorithm during the filter-scan process reads and examines the points one at a time. Each time a new point is read, is compared against the current skyline list. If its dominated by any point is discarded, otherwise is inserted in the skyline list and algorithm checks its termination trigger. If the current threshold P_{stop} is smaller or equal than the point's f_{min} value, then the algorithm terminates and returns the set of skyline points. This termination condition guarantees that all later examined data points should not be part of the skyline list, avoiding this way scanning the entire dataset.

For algorithm to compute the skyline, the values of the dataset are needed to be normalized in the range of [0,1] inclusive. Since this is not applicable in many cases the author suggest as a solution to normalize the values of the dataset as $f(p.d_i) = (p.d_i - min_i)/(max_i - min_i)$, where min_i is the minimum coordinate value on the i-th dimension and max_i is the maximum coordinate value. To demonstrate the algorithm the house-metro station dataset is normalized as sawn in **Table 8.** The first point of the sorted dataset is H1. At this point, before point H1 is read, $\underline{p_i}$ and P_{stop} are undefined. Since the set of skyline points is empty H1 is inserted in it. Values $\underline{p_1}, \underline{p_5}$ and P_{stop} are calculated since a skyline point was found. The new $\underline{p_1}$ value is 1, which is the largest value among the two coordinate values of the point and $p_{stop} = \underline{p_1}$ since it's the only $\underline{p_i}$ value due

to the only one skyline point. Next point in the dataset is H6 which is a skyline point because it is not dominated by H1. Because of the insertion of H6 in the skyline list $\underline{p_2}$ is set to 1 since it is the largest coordinate among the two of the point H6. P_{stop} remains 1 since the new $\underline{p_i}$ value is not smaller than the old one. Next point in the dataset is H8. It is not dominated and thus is a skyline point which triggers the computation of the $\underline{p_3}$ value that equals with 0,785714. The P_{stop} value is now set to 0,785714 also, since the value $\underline{p_3}$ was smaller than the current P_{stop} value. Next point in the dataset is H9 with a $\underline{p_4}$ value equals with 0,6. Since $\underline{p_4}$ is smaller than the current P_{stop} value, P_{stop} is set to 0,6. Next point is H7. It's $\underline{p_5}$ value is 0,2 since is the biggest value among the two coordinate values of the point, which also triggers the altering of the P_{stop} value to 0,2 since the new value is smaller. Next point is H11, for which the f_{min}(H11) value is bigger than the current P_{stop} value which terminates the algorithm and returns the list with the skyline points. It is observed that were processed only points that were actually skyline points and the rest were discarded saving unnecessary computations. On the downsides of the algorithm is that its performance is affected by data distribution and high dimensionality, since the pruning power of the *stop object* is limited. Additionally, because the dataset is based on a fixed ordering for each attribute, the algorithm cannot be used for arbitrary preference specifications. The advantage of the algorithm is that it can stop efficiently before the complete dataset is readied.

House	Price	Distance	f_{min}(h)	Sum(h)	$\underline{p_i}$	P_{stop}
H1	0	1	0	1	1	1
H6	1	0	0	1	1	1
H8	0,067	0,786	0,067	0,853	0,786	0,786
H9	0,600	0,071	0,071	0,671	0,600	0,600
H7	0,200	0,143	0,143	0,343	0,200	0,200
H11	0,267	0,571	0,267	0,838	-	Stop! $F_{min}(H_{11}) \geq P_{stop}$
H10	0,267	0,929	0,267	1,196	-	
H2	0,867	0,286	0,286	1,153	-	
H3	0,400	0,357	0,357	0,757	-	
H5	0,530	0,857	0,533	1,387	-	
H4	0,800	0,643	0,643	1,443	-	

TABLE 8: PRE-SORTED DATASET

2.3.10. Summary

In general, a batch-oriented algorithm will return the complete skyline faster than an online algorithm. In contrast an online algorithm will return faster than the batch-oriented algorithm a part of the skyline but it will take much longer to compute the complete skyline. Authors in **[43, 37]** suggested a set of criteria for evaluating the behavior and applicability of a progressive algorithm.

- **Progressiveness:** A part of the final set of skyline points should returned instantly and the remaining skyline points gradually.
- **Absence of false negative**: The algorithm, given enough reasonable time, should eventually produce the complete set of skyline points.
- **Absence of false positives:** The points that the algorithm returns should be guaranteed to be skyline points and not a temporary skyline points that will be discarded later.
- **Fairness:** The algorithm should not favor points that are particularly good in one dimension.
- **Incorporation of preferences:** User should be able to make preferences on the order that the skyline points are returned, while the algorithm is running.
- **Universality:** The algorithm should be easily integrated into an existing database system and be applicable to any dataset distribution and dimensionality making use of standardized technology.

In **Table 9** the algorithms are classified based on those criteria.

Algorithm	Progressiveness	Absence of false misses / Absence of temporary false hits	Absence of false hits	Fairness	Incorporation of preferences	Universality
D&C	×	√	√	√	×	√
Bitmap	√	√	√	√	×	√
Index	√	√	√	×	×	×
NN	√	√	√	√	√	√
BBS	√	√	√	√	√	√
BNL	×	√	×	√	×	√
SFS	√	√	√	√	×	√
LESS	√	√	√	√	×	√
SaLSa	√	√	√	√	×	√

TABLE 9: CLASSIFICATION OF PROGRESSIVE ALGORITHMS.

The problem of skyline computation has its roots in the fields of computational geometry and pareto-optimality and derived due to the need to retrieve pareto-optimal sets of points over datasets that do not fit directly into memory. The research approached can be distinguished on index-based and non-index-based (sorting) methods. The state-of-the-art index-based skyline algorithm is BBS. BBS is based on the R-tree to exploit the properties of nearest neighbors and identify the final skyline. The BBS algorithm is the most broadly applicable algorithm and its limitations are the curse of dimensionality derived from the R-tree. On the other hand, the state-of-the-art algorithm that does not require indexing is SaLSa. SaLSa is based on sorting to identify the most interesting points through the use of a scoring function. Those points are placed in the beginning of the dataset in order to identify the final skyline as early as possible. One of the requirements of the algorithm is to have the dataset normalized in the range of [0,1] inclusive. In addition, the algorithm is affected by the underlying data distribution. Finally, the simple BNL algorithm, due to its simple implementation and the lack of indexing and sorting mechanisms, is commonly used in numerous applications where a skyline set is needed and indexing or sorting mechanisms on top of the whole dataset are inapplicable. **Table 10** summarizes some basic properties of all the fundamental skyline algorithms.

Algorithm	Based-on	Index	D&C	Pre–processing	Sorted data	Main problem
Nested-loop join	Θ-joins [44]	×	×	×	×	Join cost
Bitmap [35]	-	Bit mapping	×	bitmaps	×	Lack of user interaction and bitmapping
Index [35]	-	Specialized B-tree	√	Index - based	Scoring function	Lack of user interaction
NN [37]	NN search and D&C scheme	Multi-dimensional index (R*-tree)	√	Index - based	Minimum distance from origin point	I/O accesses
BBS [2, 3]	NN	Multi-dimensional index (R*-tree)	√	Index - based	Minimum distance from origin point	many dominance checks / R-tree dimensionality
D&C [8]	maximal vector computation [4, 5]	×	√	× (Partial skylines can be assumed)	×	Not online/ curse of dimensionality
BNL [8]	Naive Nested-loop	×	×	×	×	Not online
SFS [41]	BNL	×	×	Sort - based	Entropy scoring function	reads all dataset / Lack of user interaction
LESS [10]	SFS	×	×	Sort - based	Entropy scoring function	Sorting / reads all the dataset
Salsa [42]	SFS	×	×	Sort - based	Min/Max Scoring function	Sorting / reads all the dataset

TABLE 10: CLASSIFICATION OF SKYLINE QUERY ALGORITHMS.

Finally, **Figure 15** illustrates the skyline algorithms via a tree structure in chronological order. The entries **[4, 5]** concern the maximal vector computation. Black lines indicate that the algorithm heavily depends or improves a previous algorithm and red dashed line indicates that the algorithm shares some general main ideas for computing the skyline.

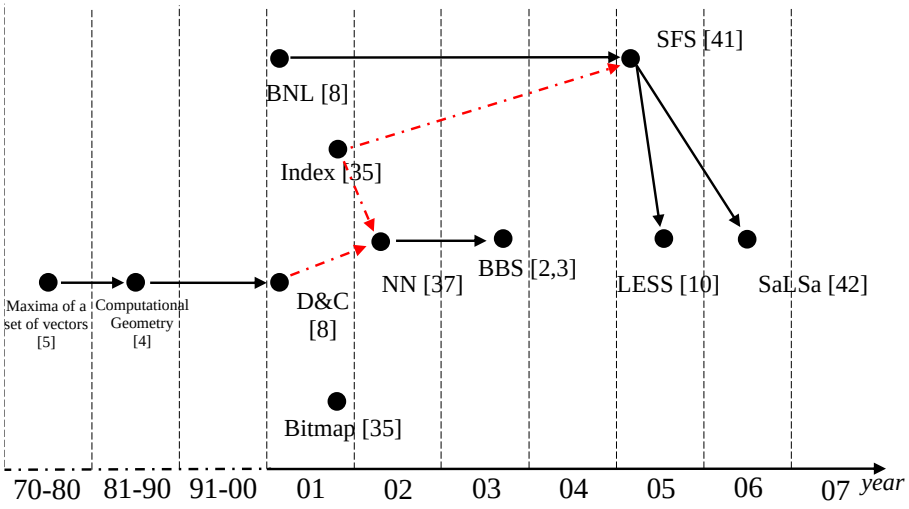


FIGURE 15: CHRONOLOGICAL ORDER OF FUNDAMENTAL SKYLINE ALGORITHMS.

2.4. Skyline Family

This section will reason about the variants of skyline queries. The main idea and notion of skyline query is maintained. Each outlined variation is applicable and can solve different aspects of a problem.

2.4.1. Constrained Skyline Queries

There are cases where a skyline query may return too many objects. This can happen if the dimensionality of the dataset is large or the dataset is anti-correlated. Additionally, users may be interested to investigate a particular subspace than the whole data space. For the previous reasons user may specify constrains on some dimensions to express those restrictions. Each constraint is typically expressed as a range along a dimension of the dataset. The constrained skyline queries are very useful in skyline maintenance in the presence of point deletions or insertions.

For this type of problems, a general variant of the skyline queries are the *constrained skyline queries* **[2, 3]** In this type of queries users are interesting in finding the skyline points of a subset of the original dataset, which satisfies one or more constraints. Given a set of constraints, a *constrained skyline query*, will return the most "interesting" points of the dataset defined by these constraints. For example, the user may be only interested in "interesting" houses in the distance range from the metro-station of 400 to 1250 and price range from 100 to 1500. For the house-metro example the *constrained skyline query* will return points H8, H11, H3, H2 **[Figure 16]** that are enclosed in the shaded region and are skyline points in that region. Point H4 which also belongs in the region will be discarded since it is dominated by H11 and H3.

Definition 3: Constrained Region
Given a d-dimensional dataset Ds a constrained region $C=\{c_1,c_2,\ldots c_d\}$ is determined by *d* sub-constraints c_i where each one expresses a range along each dimension of the dataset. That is $c_i=\{c_i min, c_i max\}$ where $c_i min$ and $c_i max$ are the minimum and maximum range restriction values on the i-th dimension.■

Definition 4: Constrained Skyline

Given a dataset Ds and a constrained region C⊆Ds, a constrained skyline will contain all the points p∈C (p.di∈ c_i , ∀ i ∈[1,d]) where ∀p,r∈C, ∃ j∈[1,d] such that p.dj<r.dj and ∀ i ∈[1,d]-{j}:p.di ≤ r.di■

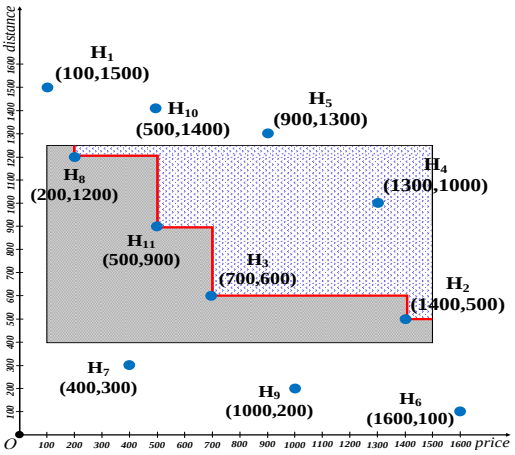


FIGURE 16: CONSTRAINED SKYLINE.

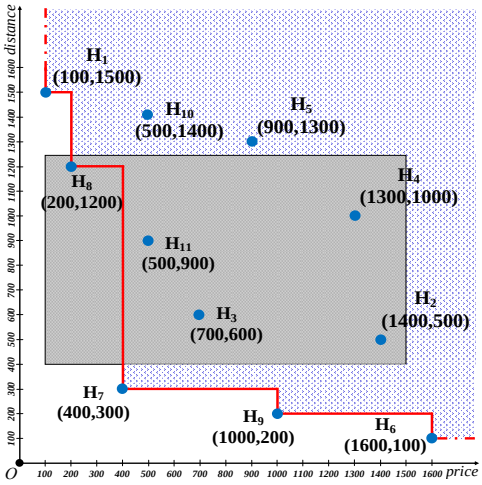


FIGURE 17: SKYLINE WITH CONSTRAINS.

Another type of queries with similar name that might confuse the reader, are *skyline queries with constraints* **[45].** This type of queries, given a set of constraints, returns the computed skyline set of the whole dataset restricted by the constraints that were placed. For the house-metro example and the constraints mentioned above, the *skyline queries with constraints* computes the skyline of the whole dataset and then applies the constraints to the retrieved skyline set and returns only the point H8 **[Figure 17]**.

In general, a *constrained skyline query* is computed over the restricted dataset by the constraints that were placed, while the *skyline query with constraints* is computed over the whole dataset and then the resulted set is restricted by the constraints. Thus, the results of both types of queries will be different (in the majority of cases) for the same dataset.

2.4.2. Dynamic Skyline Queries (DSQ)

A Dynamic skyline query is a variation of the original skyline query, which was first introduced in **[2, 3].** In this type of queries the *dynamic coordinates* of each point are given by a set of *distance (dynamic) functions* that are based on the distance between a given query/reference point q and a point p of the original dataset. The term *original space/dataset* refers to the original d-dimensional space/dataset and equivalently the term *original coordinates* to the coordinates of a point in the original space. The produced data space that occurs from the distance functions and the query point will be called *dynamic space* and the coordinates of a point in it, *dynamic coordinates*.

A *dynamic* skyline query of a d-dimensional data space DS specifies a new *d'*-dimensional data space DS' based on the original space and depicted as an inner coordinate system. To achieve this transformation specifies m (m≤d) dimension functions *f.* Each function takes as parameters one or more original coordinates of each point and maps them in a new single dynamic coordinate. That is, each point p of the original d-dimensional data space is mapped to a new d'-dimensional point $p' = (f_1(p), \ldots, f_{d'}(p))$ where each f_i is referred as a distance function. Then the dynamic skyline applied on DS with respect of functions f_i specified by a query point q returns the original skyline of the new transformed d'-dimensional space DS'.

To simplify the definition of the dynamic skyline it is assumed, without loss of generality, that DS and DS' have the same dimensionality (d=d'). Additionally for a given query point q each distance function is defined as the obsolete distance, of the i-th dimension's value of point p of the dataset DS from the i-th dimension's value of query point q, $f_i(p)=|q.di-p.di|$.

Note that dynamic skylines can have a more general class of distance functions such as Euclidian distance. In addition, they can be employed in conjunction with constrained and ranked queries (by placing weights on dimensions). An example, is the case where the absolute distance functions can receive different weights and the result of distance functions is constrained by a threshold value, i.e. find the top-3 houses within 1km given that the price is twice as important as the distance, where k is specified by user.

Definition 5: Dynamic dominance.
Given a dataset Ds, a query-reference point q in the workspace and two points p, r ∈ Ds, point p dynamically dominates point r with regard to the query point q, denoted as $p \prec^q r$ if and only if ∃ j∈[1,d] such that |q.di - p.dj|<|q.di - r.dj| and ∀ i ∈[1,d]-{j}: |q.di - p.dj|≤|q.di - r.dj| ■

Definition 6: Dynamic skyline.
Given a query-reference point q in the workspace , the dynamic skyline set of Ds with regard to the query point q, denoted as $S_{DS}{}^q$, consists of the points of the dataset that are not dynamically dominated by any other point. That is, $S_{DS}{}^q$ ={p∈DS|∄r∈DS:r $\prec^q$ p} ■

House	price (in thousand €)	Coordinate X	Coordinate Y
H1	100	+900	+1200
H2	1400	+300	+400
H3	700	-360	+480
H4	1300	+600	-800
H5	900	+500	-1200
H6	1600	+60	-80
H7	400	+240	+180
H8	200	-960	+720
H9	1000	-192	+56
H10	500	-1120	-840
H11	500	-720	-540

TABLE 11: 3-DIMENSIONAL DATASET OF THE HOUSE-METRO STATION EXAMPLE.

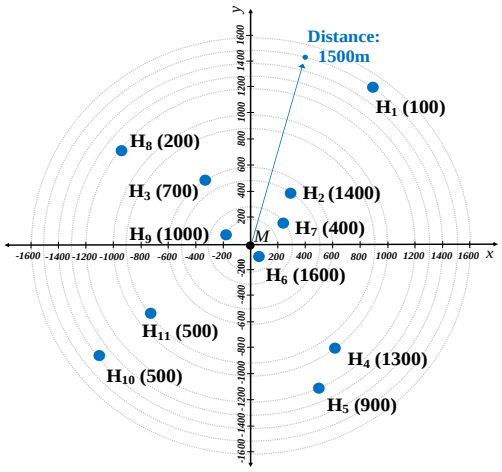


FIGURE 18: INITIAL POSITION OF HOUSES AND THEIR PRICES IN A COORDINATE SYSTEM WITH ORIGIN POINT THE METRO STATION.

In a dedicated example assume that the 2-dimensional dataset of the house- metro station example (**Table 1**) was calculated dynamically from a previous 3-dimensional dataset (**Table 11**) that had as attributes the price of each house and it's position (X,Y) in a 2-dimensional map with origin point O(0,0) the metro station's position **Figure 18**. In the case of this example and its needs, the 3-dimensional dataset is projected in a 2-dimensional one by using as a query point the position of the metro station and as distance functions the functions $f_1(H)$=(H.d1) and $f_2(H) = \left(\sqrt{(q.d_2 - H.d_2)^2 + (q.d_3 - H.d_3)^2}\right)$. This way the relative coordinated position of a house with respect the metro station is converted to the Euclidean distance of the house from the metro station (with price attribute intact) as shown in table (**Table 1**).

As with the original skyline, for the BBS to compute a dynamic skyline query, it processes the (leaf/intermediate) R-tree entries in ascending order of their *mindist*. In this case the *mindist* of a point (leaf entry) from the query point q is computed as, $f(H) = \left(\sqrt{(H.d_2 - q.d_2)^2 + (H.d_3 - q.d_3)^2} + H.d_1\right)$. The mindist of an MBR with range ([e.$d_1$min,e.$d_1$max][e.$d_2$min,e.$d_2$max][e.$d_3$min,e.$d_3$max]), from the query point q, is computed as the mindist([e.$d_1$min,e.$d_1$max][e.$d_2$min,e.$d_2$max],(q.d_1,q.d_2)) + e.$d_3$min where the first term is the mindist between the query point and the lower-left corner of the 2D rectangle [*e*.$d_1$min,*e*.$d_1$max] [*e*.$d_2$min,*e*.$d_2$max].

In a more general example, it is assumed that the user needs to find the dynamic skyline of the house-metro station dataset DS. The dynamic functions that will be used are the obsolete distances of points in the dataset from the specified query point and thus points of DS are mapped in the new space as shown in **[Figure 19],** with the same dimensionality as the original space (d=d'). In detail points H1, H2, H3, H6, H7, H8, H9, H10, H11 are projected to points H1', H2', H3', H6', H7', H8', H9', H10', H11' respectively **[Table 12]** with regard the query point q and the dimension functions $f_1(H)=|q.d1 - H.d1|$ and $f_2(H)=|q.d2 - H.d2|$. The dynamic skyline for the selected query point contains houses H3', H11' which are essentially points H3, H11.

House	price (in thousand €)	Distance (m)	Dynamic price (in thousand €)	Dynamic Distance (m)
H1	100	1500	1500	1500
H2	1400	500	1400	1100
H3	700	600	900	1000
H4	1300	1000	1300	1000
H5	900	1300	900	1300
H6	1600	100	1600	1500
H7	400	300	1200	1300
H8	200	1200	1400	1200
H9	1000	200	1000	1400
H10	500	1400	1100	1400
H11	500	900	1100	900

TABLE 12: ORIGINAL AND DYNAMIC DATASET.

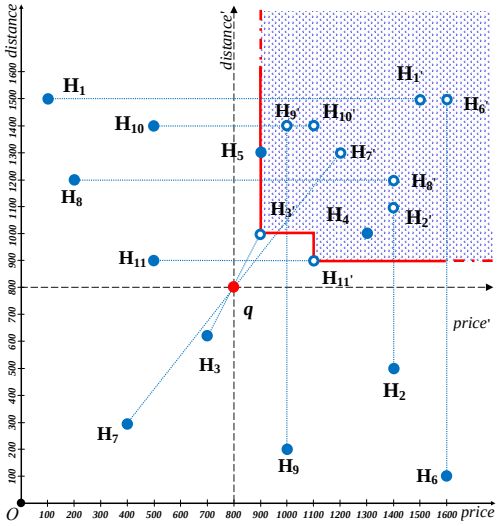


FIGURE 19: DYNAMIC SKYLINE.

A DSQ query can be seen as a query from the buyer's perspective, by identifying the houses that are most interesting to him.

2.4.2.1. Spatial Skyline Queries (SSQ)

The spatial skyline query (SSQ) **[46, 47]** can be considered as a more restricted special case of the dynamic skyline queries. It considers multiple query points at the same time and relies on the existence of a multi-dimensional Euclidean space to derive geometric bounding structures, such as convex hull and Voronoi diagram to reduce the search space. Given a dataset DS and a set of query points Q, a Spatial Skyline Query retrieves those points of DS, which are not spatially dominated by any other point in DS with respect to Q. Specifically, a point $p \in DS$ spatially dominates a point $r \in DS$ with respect to Q, if and only if p is closer to at least one query point $q \in Q$ as compared to r and has in the best case the same distance as r to the rest of the query points, i.e. no other object is closer to all the given query points simultaneously.

Geometric notations

Convex Set: A set S of points, that exist on a plane over $\mathbb{R}$, is called convex set if and only if for any two points $p,r \in S$, the segment (line) that connects them resides entirely in S (i.e. all the points of a circle or a hexagon) ■

Convex Hull: The convex hull of a set S of points over $\mathbb{R}$, is the intersection of all the convex sets containing S ■

A counter example of a convex hull would involve the dashed line in **Figure 20**. If the segment of the red line, which belongs between houses H4 and H6 was replaced be the dashed line that contains house H2, the set S would not be a convex set since the line that connects houses H4 and H6 would not reside in S.

Voronoi diagram: Given a set S of n points over $\mathbb{R}$ that exist on a plane, the Voronoi diagram of S, is the subdivision of the plane in n cells, where each cell contains only one point of S, called generator. The important property is that any point (except the generator) in a particular cell will be always closer to the point that generates this cell (**Figure 21**)■

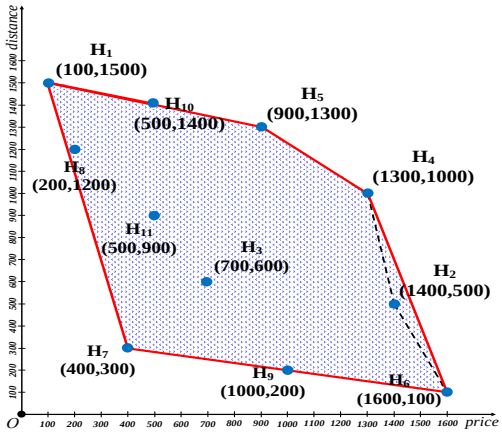


FIGURE 20: CONVEX HULL OF THE HOUSE-METRO STATION DATASET.

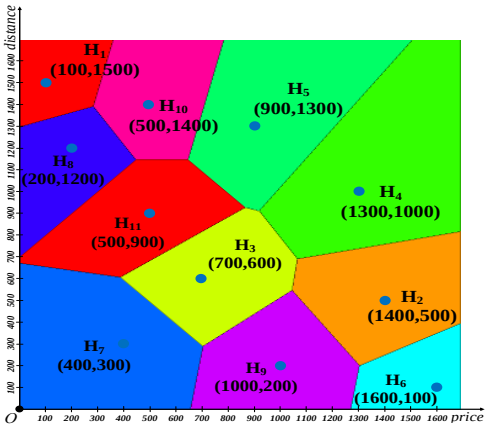


FIGURE 21: VORONOI DIAGRAM OF THE HOUSE-METRO STATION DATASET.

To reduce the search space authors give two important theorems; the spatial skyline points, which are those points, which either within the convex hull **[48]** of query points or having their own Voronoi cells **[48]** intersect with boundaries of the convex hull of query points. Also, they proposed the R-tree-based B^2S^2 algorithm and the Voronoi-based VS^2 algorithm for the *spatial skyline queries*. Both algorithms are efficient in cases where only Euclidean distances are considered as dimension functions, but their search structures are inefficient in high-dimensional and metric spaces. Additionally, authors proposed a Voronoi-based continuous VCS^2 algorithm to efficiently update a spatial skyline taking into account that the location of query point q can change. Moreover, they extended their work in **[47]** by computing the spatial skyline query in the metric space of Spatial Network Databases (SNDB), such as the road networks, in which the spatial objects are restricted in predefined locations/routes.

To demonstrate the use of spatial skyline queries, consider the example from a set of home heating oil delivery stations (data points P) where their user wants to identify a candidate subset to dispatched delivery trucks to multiple houses (query points Q). This candidate subset includes those stations that are not dominated by any other station with respect to all the houses, and hence they are the spatial skylines.

In **[49]** is proposed the Multi-Source Skyline Query (MuSSQ) in road networks where the network distance between two locations needs to be computed on-the-fly and the attributes are defined to be the shortest path length from data points to query points. In **[50]** it is proposed the *Location-Dependent Skyline Query (LDSQ)* for multi-objective distance optimization, considering a continuous changing user location (query point). In **[51]** authors consider spatial skyline computation with user preference information in addition to distances. Also, they extend the query processing algorithm to return at least *k* good objects (where *k* is a user specified number) even when the original skyline contains fewer than *k* items. In **[52]** is proposed the *Direction-based Spatial Skyline Query* (*DSSQ),* which finds the best objects by comparing them in terms of distance from a mobile user and also by considering the direction that the user moves, rather than only distance as in traditional spatial skyline queries. In **[53, 54]** authors propose Manhattan Spatial Skyline Queries (MaSSQ) and develop an efficient algorithm for spatial skyline queries using the L1 norm, also known as Manhattan distance. Readers must not associate Manhattan Spatial Skyline Queries (MaSSQ) with Multi-Source Skyline Queries (MuSSQ).

2.4.3. Reverse Skyline Queries (RSQ)

A Reverse Skyline Query **[14]** retrieves these points in the database whose dynamic skylines contain a given query point. This type of query, as opposed to the DSQ, is a query from the perspective of real estate company. For example, given the ideal preferences of potential house

buyers, as points in a two-dimensional space, the reverse skyline query can answer the question if it make sense to offer a house q (as a query point) to one of the potential buyers. The house q (becoming an origin point) will be interesting for a buyer, if it will be part of the dynamic skyline of his preferences (that represent the dataset points). Another example would be the selection of a new store's location. A reverse skyline query on a customer database, with respect to a query point q that represents the new location of the store would return those customers who are potentially interested in the new store. Then the strategy is to select the location that maximizes the number of customers.

Definition 7: Reverse skyline.
Given a dataset *Ds* in a *d*-dimensional space *D* and a query point q (q_1, q_2, ..., q_d) $\in D$, the reverse skyline query of *Ds* with regards to q retrieves the set of points $RSL_q(Ds) \subseteq Ds$ for which q is a dynamic skyline point of *Ds* with regards to all points in $RSL_q(Ds)$, that is, $RSL_q(Ds) = \{p \in Ds \mid \nexists r \in Ds: r \prec^q_{Ds} p\}$. The points in $RSL_q(Ds)$ are called reverse skyline points of *Ds* with regards to q■

Definition 8: Global Domination
Given a dataset *Ds* in a *d*-dimensional space *D,* a query point q ($q.d_1$, $q.d_2$, ..., $q.d_d$) $\in D$ and two points p($p.d_1$, $p.d_2$, ...,$p.d_d$) ,r($r.d_1$, $r.d_2$, ..., $r.d_d$) $\in D$, point p will *globally dominate* r with regard to the query point q (denoted as $p \prec^q r$) if $\forall i \in \{1, \ldots, d\}$: $\{ (p.d_i - q.d_i)(r.d_i - q.d_i) > 0$ and $|p.d_i - q.d_i| \leq |r.d_i - q.d_i|\}$ and $\exists j \in \{1, \ldots, d\}$: $|p.dj - q.dj| < |r.dj - q.dj|$■

Definition 9: Global Skyline
Given a dataset *Ds* in a *d*-dimensional space *D* and a reference point $q \in D$, The global skyline of a point q, GSL(q), will contains the points which are not globally dominated by another point according to q■

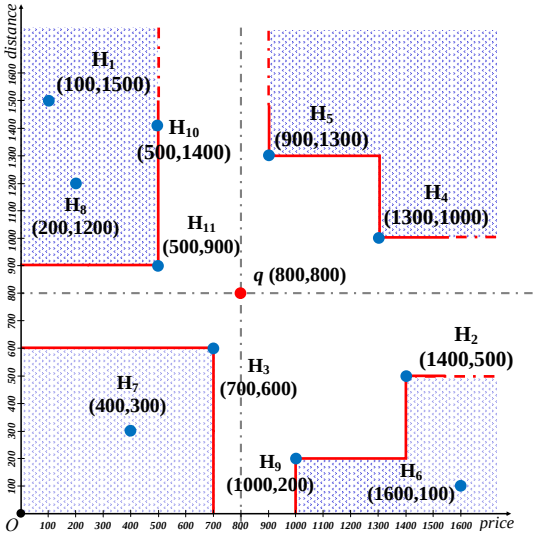


FIGURE 22: GLOBAL SKYLINE AND RANGE QUERIES.

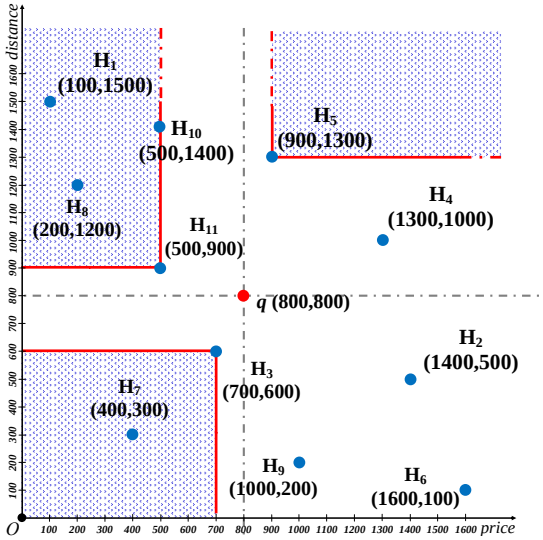


FIGURE 23: REVERSE SKYLINE.

To compute the reverse skyline (RSL) of the house-metro station example, with regards the query point q (800,800), it is first needed to compute the global skyline GSL(q) as shown in **Figure 22.** As illustrated, the GSL(q) will contain the (reverse skyline candidate) points H_{11}, H_7, H_5, H_4, H_9 and H_2. The resulted reverse skyline **Figure 23** will eventually contain the points H_5, H_7, H_{11}. The rest of the points are discarded because, in order for a reverse skyline candidate p to be a reverse skyline point must not exist any point q in the GSL that is (strictly) better, in terms of distance from q, on all dimension simultaneously.

With the existing introduced algorithms, in order to compute the RSQ of a dataset DS, given a query point q, it is needed to examine all the points of DS by performing a dynamic query (e.g. using BBS) for each point to find the points that have q as part of their dynamic skyline. The points that will be retrieved would be the reverse skyline set. A first optimization in this approach would be to stop processing the dynamic skyline of a point when q is already identified as a skyline point since there is no need to compute the entire skyline. To further optimize the identification of Reverse skyline point authors proposed two algorithms *Branch-and-bound algorithm* (BBRS) and *Reversed Skyline Search with Approximation* (RSSA). BBRS is an improved customization of the original BBS algorithm and uses a Multidimensional index (e.g. R-tree). Its goal is to process the reversed skyline of a query point q without applying a space transformation. To achieve this, it retrieves the proposed Global Skyline GSL(q) that returns a small subset of the dataset as candidates for RSQ (this subset is still a superset of RSQ), which essentially reduces the search space for the reverse skyline computation. Algorithm RSSA computes the dynamic skyline for each point of the dataset and uses an accurate pre-computed approximation of the skyline in a filter-refinement step to compute the reverse skyline. Along with the RSSA algorithm, authors proposed an optimal algorithm to compute approximations for two-dimensional skylines and a greedy algorithm for higher dimensions. The basic idea of the approximation scheme is to pre-compute the dynamic skyline of each point of the dataset and select a fixed number *k* of K_{max} dynamic skyline points ($k \leq K_{max}$).

Some additional applications where the reverse skyline can be applied is the case when needed to identify customers that would be interested in a particular product by exploring the dominance relationships between other competitor's products, with respect of the user preferences. The reverse skyline can also be applied in situations such as environmental monitoring, where several sensors are deployed to monitor the area and report data such as temperature and humidity.

In **[55]** authors try to answer the so called why-not questions in reverse skyline queries To answer this type of question we need to find why a point does not belong in the reverse skyline and what actions are needed to be performed (to the query point but also to the why-not point) to be part of the reverse skyline by incurring only minimum changes to both.

2.4.4. Group-by and Join Skyline Query

In this section are introduced the *Group-by Skyline queries* **[3]** and *skyline queries over joins* **[56]**.

2.4.4.1. Group-by Skyline Query

To illustrate a Group-by skyline query **[3]** example based on the initial house-metro station dataset, a third attribute is inserted into the original dataset **Table 1** (without altering any of its values) represents the number of bedrooms that each house has **(Table 13).** This way a potential buyer can find individual skylines depending on the number of bedrooms. That is to group the houses by the number of their bedrooms and then compute the skyline of each group. In this case the cardinality of distinct values of bedrooms will be equal to the number of individual skylines that will be found. In **Figure 24** the individual skylines of each group based on the number of bedrooms are illustrated.

House	price (in thousand €)	Distance (m)	No. of bedrooms
H1	100	1500	1
H2	1400	500	3
H3	700	600	2
H4	1300	1000	3
H5	900	1300	2
H6	1600	100	3
H7	400	300	1
H8	200	1200	1
H9	1000	200	2
H10	500	1400	1
H11	500	900	1

TABLE 13: HOUSE-METRO STATION DATASET WITH NO. OF BEDROOMS.

House	price (in thousand €)	Distance (m)	No. of bedrooms
H1	100	1500	1
H7	400	300	1
H8	200	1200	1
~~H10~~	~~500~~	~~1400~~	~~1~~
~~H11~~	~~500~~	~~900~~	~~1~~
H3	700	600	2
~~H5~~	~~900~~	~~1300~~	~~2~~
H9	1000	200	2
H2	1400	500	3
H4	1300	1000	3
H6	1600	100	3

TABLE 14: GROUP-BY SKYLINE.

To give a formal definition of the Group-by dominance property and the Group-by skyline it is needed to define the following:

Given a relational table instance *DS* (dataset), in a d-dimension space with equal numeric attributes and a schema $A= (A_1, A_2,\ldots A_d)$, the notation $p[A_i]$ represents the value of a tuple *p* in the attribute A_i. Additionally, given a set $G \subset A$ of attributes of *DS* that will be used for grouping and an instance *g* of *G* (i.e. one distinct value from the total values of number of bedrooms), *DS(g)* is defined as the set of tuples of *DS* that belong to the group instance of g. That is: $DS(g) = \{p \in D | \forall A_i \in G, p[A_i] = g[A_i]\}$

Definition 10: Group-by Dominance
Given a set S ⊂ A (S∩G=∅) which this time contains the skyline attributes (that will be checked for dominance), a tuple p dominates another tuple r with respect of *S*, (denoted by $p \succ_s r$) if and only if $\exists\ A_j \in S$ such that $p[A_j] < r[A_j]$ and $\forall\ A_i \in S - \{A_j\}$: $p[A_i] \leq r[A_i]$ ■

Definition 11: Group-by Skyline
Eventually the Group-by skyline query will contain all the tuples *p* that are not *Group-by dominated* by any other tuple *r* with respect of *S* and that is: $\Psi(DS, S) = \{p \in DS | \nexists r \in DS, r \succ_S p\}$■

Summarizing a group-by skyline query Q= (G, S), with G representing the grouping attributes and S the skyline attributes (S∩G=∅), computes the skyline result set ψ(DS(g),S) for each group instance g defined on G and the overall query result can be represented as Q (DS).

Based on the dataset *DS* of **Table 13**, to find the group-by skyline with respect to the *No. of bedrooms*, the grouping attributes are defined to be G= {No. of bedrooms} and the skyline attributes S= {Price, Distance} (S∩G=∅). The dataset *DS* of **Table 13,** is partitioned into groups (**Table 14**) based on G and then the skyline tuples of each group are computed with respect of S.

A naïve approach to process a Group-by skyline is to create a separate R-tree for each one of the distinct values of bedrooms. Each R-tree will contain the corresponding house entries with their two remaining attributes, depending on the number of bedrooms (grouping attribute), and then an original BBS algorithm on each tree will be invoked. Nevertheless, this approach is inefficient since the performance of queries when all attributes are involved is compromised as it may be needed to maximize or minimize the grouping attribute. A more efficient approach, which operates on the R-tree that indexes all the attributes is achieved with a variation of BBS **[2].** This variation stores the already found skyline points for every group, in a secondary *(d-1)-dimensional* (in this case) R-tree and maintains a heap with the visited entries. The sorting measure that is used is based only on the *d-1* remaining attributes (without the group-by attribute). The dominance check of a retrieved point, from the original R-tree, is performed on the corresponding by its group R-tree and is inserted in it only if it is not dominated by any of the existing points. Dominance checks for intermediate entries (regions) are more complicated because it is likely to contain hotels of several classes.

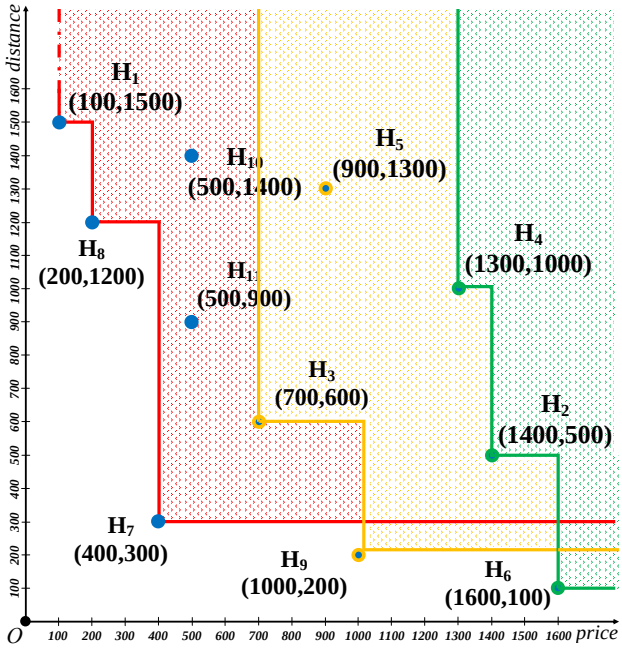


FIGURE 24: GROUP-BY SKYLINE.

Authors in [57] proposed the aggregate skyline query which combines the skyline and group-by queries. Essentially, the aggregate skyline is the set of groups not dominated by other groups. The various groups are defined based on a common property of tuples. In addition, authors discussed the differences between the efficiency of the aggregate skyline query processing in relation to the sequential execution of the skyline and group by query.

In [58, 59] authors studied the problem of identifying the k-tuple skyline groups. In this problem authors try to identify groups of k tuples that are not dominated by other, equal sized, groups. To compare the groups, each group is associated with an aggregate vector which is computed based on a common aggregate function such as SUM, MIN, MAX. The aggregate values of vectors are computed based on all the attributes of all tuples in a group. A naïve approach to compute the k-tuples skyline groups is to compute the aggregate functions for each k-tuple combination and then invoke a traditional skyline algorithm to identify the skyline groups.

2.4.4.2. Skyline Queries Over Joins

Most of the existing work discusses the computation of skyline queries over data that are stored in a single table. In [56] authors discuss the case where data are stored in multiple tables and thus is required to perform join operations among them to compute the final skyline and propose efficient methods to share the join processing cost with skyline computation cost. More specifically assuming the existence of two (or more) tables (which have one common join attribute) and apply a join operation on them, there is a case that may appear new skyline points that are not in the skyline of the individual tables. Based on this observation a naïve approach would be to compute the join of the two tables and produce a new table that contains the joined records. Afterwards apply a skyline query to the derived table to compute the skyline. The problem that may arise in this case is the potential increment of the computational cost of skylines on the joined table due to its increased cardinality and dimensionality. As a solution authors proposed a sort-merge join approach where they group the tuples in three groups according to the values of join attributes and based on whether or not are local skylines in their group and skyline points in the whole table. This involves a first phase of pruning from each table which is achieved with the use of an R-tree. Afterwards each tuple in each group is sorted based on its join attribute value. The next step involves a third (in this case) table which will host the join operation. Each group is inserted individually and merged with the existing tuples by additionally performing a dominance check for each tuple to compute final skyline of the join relation.

2.4.5. Top-k Skyline Query

Top-K skyline queries (or ranked skyline queries) were proposed in **[2, 3]** and return the *K* "most interesting" skyline points of a given dataset, based on a monotone preference function *f*. The user specifies the parameter K, which represents the number of points to be reported and the monotone preference function *f* based on the weighting that wants to apply over the attributes. The query will return the K points of the dataset with the minimum (or maximum) score according to the function *f*. To demonstrate this with an example assume that K=3 and the preference function is f(x)=x+2y (i.e. a lower *distance (y)* is more important than *price (x)* to the user). The Top-3 points of the house-metro station dataset that will be returned are {(H_7, 1000), (H_9, 1400), (H_6, 1800)}. This type of queries can be efficiently solved with BBS algorithm by replacing the *mindist* function with the given preference function. In this case the algorithm will terminate when exactly K points have been retrieved.

2.4.6. Thick Skyline Query

A Thick Skyline **[60]** extends the conventional skyline by returning the conventional skyline points and additionally their nearby non-skyline neighbors that exist within ε–distance, which are similar but not as good as the skyline points. This approach can help the user in cases of nearest neighbor search where the cardinality of the dataset is high and the points of the dataset forms groups. In their work the author extend the concept of *skyline* to *generalized skyline* by adding a user-specific constraint, defined as the *ε-neighbor* of any skyline point, into skyline search space. A *Thick skyline is* composed by a subset of the generalized skyline points.

Definition 12: Generalized Skyline.
Given a d-dimensional dataset DS and a set SL={s_1, s_2, s_3,… }, which contains the conventional skyline points of DS, the generalized skyline GL is consisted from the conventional skyline points and additionally the non-skyline points that exist in their vicinity within ε-distance. That is GL=SL∪{p|p∈DS^p∈s_i+ε, ∀i, 1≤i≤d}■

Definition 13: Thick Skyline.
Given a d-dimensional dataset DS, a thick skyline is composed by the skyline points of the generalized skyline (named *dense skyline* points), that have in their vicinity (defined by ε) another (strictly) skyline point(s), and additionally the skyline points of the generalized skyline (named *hybrid skyline* points) that have in their vicinity another skyline point(s) and some non-skyline points. Thus, a thick skyline contains all the skyline points of the generalized skyline except of those that do not contain any other point in their vicinity (*outlying skyline* points*) as defined by the* generalized Skyline ■

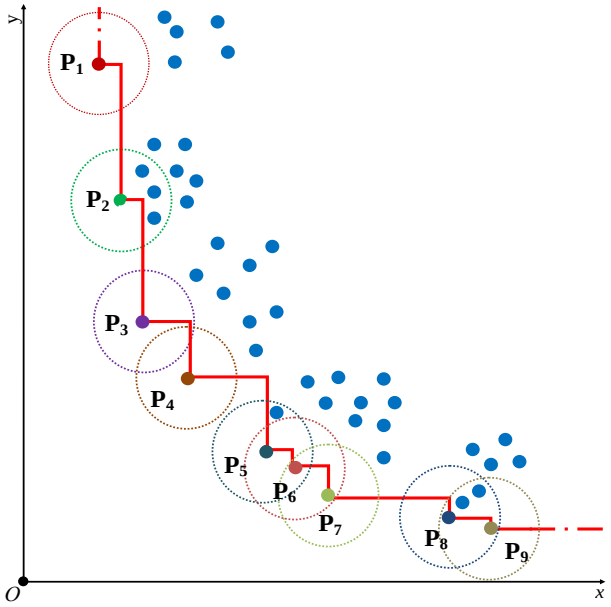


FIGURE 25: DENSE, HYBRID AND OUTLYING SKYLINE POINTS.

Figure 25 shows the main differences between the *dense*, *hybrid* and *outlying skyline points* in a plane that is consisted of random points, because in the house-metro station dataset this would not be obvious. Point p_1, p_3, p_4 are outline skyline points since are skyline points but they do not contain any point in their vicinity. Point p_7 and p_6 are dense skyline points since are skyline points and contain another skyline point in their vicinity. Point p_2, p_8, p_9 and p_5 are hybrid skyline points since are skyline points and contain other non-skyline points in their vicinity. Additionally, point p9 and p8 contain another skyline point in their vicinity.

Authors proposed three algorithms, *Sampling and Pruning*, *Indexing and Estimating* and *Microcluster-based* algorithm for mining thick skylines under three typical scenarios. The first scenario concerns a single file to represent the dataset where the sampling and pruning technique exploits statistics from the database, such as order and quantile in each dimension, to identify the thick skyline points by comparing the points of the dataset according to the defined *Strongly Dominating Relationship*.

Definition 14: Strongly domination relationship.
A point p strongly dominates a point q (denoted as $p \triangleright q$), if $p+\varepsilon$ dominates q. That is $\forall i \in [1,d]$, $p_i+\varepsilon \leq q_i$ and $\exists j \in [1,d]-\{j\}$, $p_i+\varepsilon < q_i$ ■

The second scenario concerns a general index structure such as *Index* algorithm **[35]** where points are partitioned in lists ordered by their minimum coordinate and compared in a similar order. The final scenario concerns the partitioning of the dataset into microclusters based on the CF-tree structure **[61].** Then the algorithm follows a similar approach as BBS **[2]** to identify the desired points using bounding and pruning techniques. In each case the thick skyline performs approximate selections, as it employs approximate measures and increases the size of the final result set (compared to the conventional skyline) that is returned to users.

2.4.7. K-representative and Distance-based Representative Skyline Queries

K-representative skylines points (top-k RSP) were proposed in **[62]** to identify a set of k skyline points that maximize the total number of (distinct) points dominated by one of the *k* skyline points. This type of query was proposed to allow users to have a good approximation (returning few but representative skyline points) of the final skyline and let them make a good and quick selection when the skyline is consisted from too many points. Authors also developed an efficient, scalable, index-based randomized algorithm. Authors in their implementation employed the *BBS* **[2, 3]** algorithm and the *FM probabilistic counting algorithm* **[63].** The FM algorithm is a bitmap based

algorithm that can efficiently estimate the number of distinct elements (data points) dominated by a skyline point, overcoming multiple-domination counting.

2.4.7.1. K- Representative Skyline (Top-k RSP)

Given a dataset Ds and an integer k, ∀p∈Ds, D({p}) is denoted as the set of points in Ds that are (strictly) dominated by p. For a set S of data points, with S⊆DS, D({S}) denotes the set of points each of which is strictly dominated by a point s∈S. The set K of the k-representative skyline points will contain k skyline points that Maximizes |D(K)|■

The problem of identifying the *K*-representative skyline is known to be NP-hard **[62]** in 3 or higher dimensional space. This approach is scale invariant but cannot be considered stable since adding a non-skyline point may alter the final k-representative skyline set. Top-k RSP can be transformed in the *maximum coverage problem* **[64]** and solved approximately by the author's proposed greedy heuristic.

2.4.7.2. Distance-based Representative skyline

Authors in **[65]** proposed the *distance-based representative skyline,* which is an alternative solution for the problem of *k-representative skyline* points (referred as *max-dominance representative skyline* in this work) where they redefined it. The reasoning was that the set K of k points returned by the *k-representative skyline* can turn out not to be representative, because the produced points may belong to the same cluster or the set K fails to represent the extreme points. From the authors perspective a good representative skyline should have for every non representative skyline point, a nearby representative. Therefore, in their work they defined the problem of identifying the *k-representative skyline points* as the set of k points that minimizes the distance between a non-representative skyline point and its nearest representative. The proposed approach it is not a scale invariant, as the previous approach (*k-representative skyline*), since it is based on distances. As opposed, it can be considered to be stable since by adding a non-skyline point in the dataset will not change the final representation (due to the initial algorithm construction).

In this approach it is considered that the data space is normalized in the range [0, 1]**.** The distance-based representative skyline can be an optimal solution for k-*center problem* **[66]** of the full skyline. Except from the *distance-based representative skyline* authors introduced the concept of *representation error* of *K*, denoted as *Er*(*K, S*) to quantify and check the quality of the representation K of the full skyline S of the dataset Ds, by the k identified representatives. This is achieved by checking the maximum of all distances, between any of the non-representative skyline points in the set *S−K* and their nearest representative in *K*.

Definition 15: Distance-based Representative Skyline.
Given a dataset Ds, its skyline set S and an integer value k, the distance-based representative skyline K of Ds is consisted of k-skyline points of S that minimizes the *representation error Er(K,S)*■

For the 2-dimensional space authors developed a dynamic programming algorithm that optimally finds a solution in polynomial time. For 3-dimensional spaces and higher authors propose a *2-approximate* **[67]** polynomial algorithm and prove that the problem is still NP-hard **[62]**. The algorithm can quickly identify the *k* representatives without extracting the entire skyline by utilizing a multidimensional access method (i.e. R-tree). The proposed algorithm is progressive and does not require a specific k value from the user as it continuously returns representatives that are guaranteed to be a *2-approximate* solutions at any moment, until either manually terminated by the user or eventually producing the full skyline.

The k-representative skyline gives to the user a high-level summary of the entire skyline as it returns only a few points that reflect to the contour of the final skyline and then progressively refines it (contour) by reporting more skyline points. The user may identify interesting representative points and request only the skyline points that are similar to those representatives (i.e. belong to a specific part of the contour).

2.4.8. ε-skyline

The proposed type of query claims that solves the limitations and drawbacks of the original skyline by taking into account that there was no algorithm that can simultaneously control the resulted size of the skyline, has built-in ranking for the points and weighting on dimensions. According to previous considerations authors proposed the ε-skyline [68] which allows users to control the number of output skyline points (by increasing or decreasing them depending on an appropriate ε parameter that the user defines), provides a built-in ranking system and integrates weighting factors for each dimension.

The algorithm takes as input a d-dimensional dataset (with its values normalized as on SFS [41] in [0, 1] (**section 2.3.7**), a weight vector W that will contain the weight factors W_i, (i∈[1, d]) for each dimension (if no weighting is needed all the factors will be equal to 1) and a parameter ε∈[-1, 1]. The dominance property is relaxed according to the ε parameter. To manage the dominance relations, authors defined some additional properties such as *irriflexivity*, *loose transitivity* and *loose asymmetry*. The weights that the user inserts are incorporated in the dominance comparisons. For the built-in ranking system to work every point *p* in the dataset has a corresponding ε-max value which represents the largest value of ε, which makes *p* to be a skyline point. Thus, the points have a natural order based on ε-max value. This ordering can be used to place top-k ε-skyline queries.

Definition 16: *ε*-domination.
Given a d-dimensional dataset *DS*, a weighting vector W={W_i |i∈[1, d],0<w_i<1}, a parameter ε∈[-1, 1] and two points p,r∈DS, p ε-dominates r, denoted with p $\prec^{\varepsilon}$ r if and only if ∃ j∈[1,d] such that p.dj<r.dj and ∀ i ∈[1,d]-{j}: p.di *w_i ≤ r.di *w_i+ε■

Definition 17: *ε*-Skyline.
The *ε*-Skyline of a dataset *DS* contains all the points p∈DS that are not ε-dominated by any other point on the dataset■

In their work, **[68],** authors proposed two algorithms, ε-SFS which is progressive and based on the SFS [41] algorithm and IFR (index-based Filter-Refinement) algorithm which uses an index structure such as an R-tree and a filter-refinement framework.

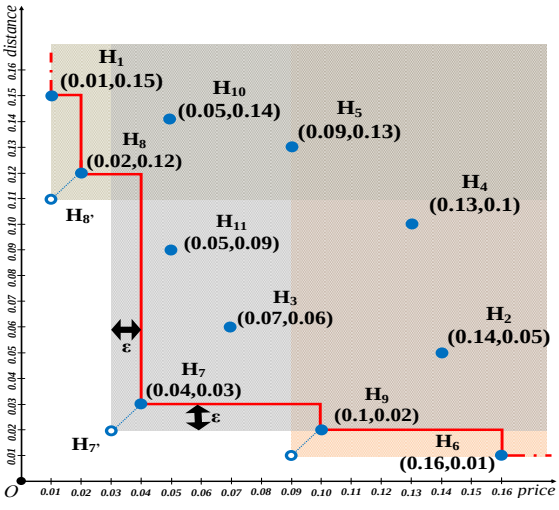


FIGURE 26: DOMINANCE REGION OF H_7' WITH E=0.01 .

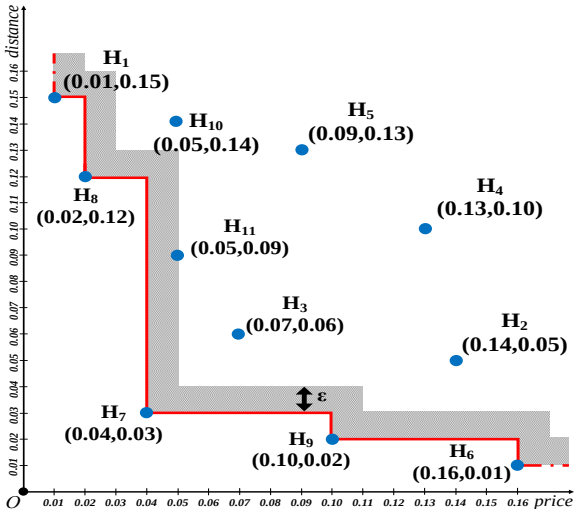


FIGURE 27: E-SKYLINE WITH E=-0.01 .

An ε-skyline can monotonically vary from an empty set to the whole dataset depending on the value of ε. An ε-skyline with ε=0 represents the conventional skyline. An ε-skyline with value ε=-1 will return the whole dataset and with value ε=1 the empty set. More generally the case of an ε-skyline, with ε>0 is shown in **Figure 26**. In this case ε=0.01 and as shown the dominance region

of (i.e.) H_7 will be visualized as the point H_7 was moved to the location of H_7' to fulfil the ε-domination. This way point H_7 ε-dominates point H_9 which for that reason will not be in the final skyline set. Similar H_8 ε-dominates H_4 and H_9 ε-dominates H_6. The final skyline set for ε=0.01 will be S= {H_7, H_8}. The case of an ε-skyline, with ε<0 is shown in **Figure 27.** In this case the ε-skyline will contain the conventional skyline points and additionally the points that are in the shaded area, as this happens with point H_{11}.

2.4.8.1. Approximately Dominating Representatives

In **[69]** authors introduced the notion of approximately dominating representatives (ADRs). The scenario has a set of n points in a d-dimensional space and a value ε>0, where it is desired to find the minimum set of points, named ε-ADR, that approximately dominate all the points of the dataset. With this approach they try to minimize the number of (skyline) points to be reported at a small loss of accuracy. The approximation is imposed by a user-defined value ε that extends the dominating region of each point. The data points retrieved by the algorithm when ε=0 (ε-ADR) are guaranteed to be skyline points. In this case the algorithm will return all the existing skyline points. In the cases of ε>0 may exist many different ε-ADRs (for a specific value ε). In this case the points returned are not guaranteed to be skyline points. An example of a case where ε>0 can be considered a dataset that contains a point that approximately dominates all others (i.e. a point very close to the origin of axes if minimization of preferences is desired). In this case the algorithm will return only this point, although it may not be a pure skyline point.

2.4.9. Enumerating and K-dominating Queries

Enumerating queries and K-dominating queries (Top-k dominating queries in general bibliography) were proposed in **[2, 3].** These types of queries do not produce skylines but can work as a measure of "goodness" in various cases.

2.4.9.1. Enumerating Queries

An enumerating query **[2, 3]** returns the set of skyline points and additionally the number of points that each skyline point *p* dominates (denoted as *num(p)*). This kind of result could be used to investigate which skyline points are more interesting by means of "number of points that they dominate". To compute the enumerating query the first step is to retrieve the skyline points of the dataset with an existing algorithm (i.e. BBS). The second step performs a query in the R-tree, for every skyline point, to count the number of points that exist in their dominance region. To avoid multiple node visits with the previous technique (since a node may be dominated by more than one skyline points), a solution is to apply the inverse procedure which is, for each non-skyline point in the dataset, perform a query in the R-tree to find the dominance regions that contains it and accordingly increase the appropriate counters of the skyline points that dominates it. As an example in the house-metro station dataset the enumeration query will return for the house H_7, num(H_7)= 6, for the house H_{11}, num(H_8)= 2 and for the house H_6, num(H_6)=0.

2.4.9.2. k-dominating Queries

A variation of the above problem (and also the predecessor of the *k-representative skyline query*) that incorporates the enumerating query (and the constrained skyline queries) is the K-dominating query **[2, 3]**. This type of query returns the *K* points that dominate the largest number of other points. The points that are returned do not necessary belong to the skyline of the dataset.

The first step to retrieve the K-dominant points is to perform an enumerating query. The query will return the skyline points and the number of points that each one of them dominates. The items that are retrieved are sorted by their descending order of the number of points that they dominate and the first K points are placed in a list. The first point of the list is the first result of the k-dominating query and it is returned to the user, removed from the list and pruned from further computations. Next is applied a local (constrained) skyline with boundaries the (exclusive) dominance region **(Figure 28)** that was defined by the point removed, to efficiently find the skyline of the dataset after the removal of the first point and identify potential candidate K-dominant points (that may outnumber points in the list). The second step of the enumeration query is applied on the newly found skyline points (if they exist and are possible candidates) and returns the number of points that they dominate. If any of the points found outnumbers the last point of the list, it

replaces it and the list is rearranged. The first point of the list will be the second K-dominating point. The algorithm terminates when it finds the K most dominant points, thus when the new points retrieved from the local skyline cannot outnumber the points in the list. For the House-metro station example a K-dominant query for K=3 will return the points {(H_7,6),(H_{11},3), (H_8,2)}. More analytically after the initial enumerating query the list will contain the points {(H_7,6), (H_8,2), (H_9,2)}. Point H7 will be the first K-dominant point and returned to the user. After removing point H_7, the local skyline point H_{11} is checked and is inserted in the list (the last point of the list is removed resulting in {(H_{11},3), (H_8,2)}. The local skyline point H_3 is also checked, after the removal of H_7, but is not inserted in the list because it does not outnumbers any point in it. Thus, the second K-dominant point H_{11} is returned to the user but the algorithm terminates **(Figure 29)** since a local skyline, by removing H_{11}, has not any candidate that may outnumber the last (and in this case the final) point of the list. So point H_8 is also returned to the user.

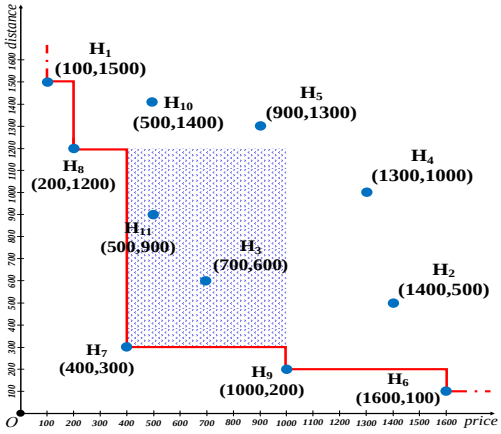


FIGURE 28: EXCLUSIVE DOMINANCE REGION OF H_7.

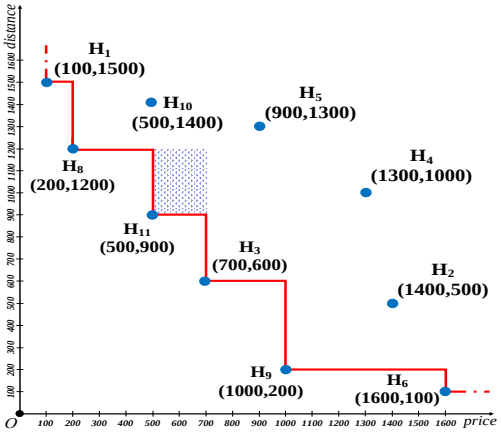


FIGURE 29: SKYLINE AFTER REMOVING H_7 (FINAL STEP OF ALGORITHM

2.4.10. k-skyband Query

A K-skyband query [3] returns the points that are dominated by at most K points with the case of K=0 representing the original skyline. A K-skyband query follows similar logic with K nearest-neighbor query with K representing the thickness of the skyline.

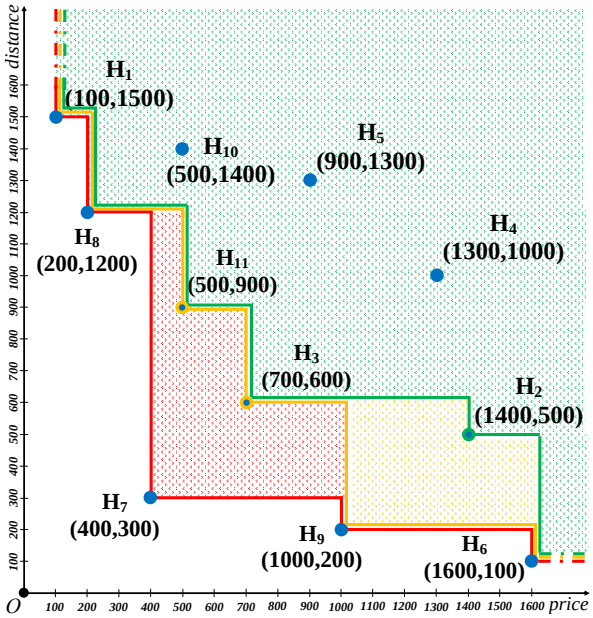


FIGURE 30: (0, 1, AND 2)-SKYBAND QUERY.

Figure 30 illustrates a 0-skyband query (red line), a 1-skyband query (yellow line) and a 2-skyband query (green line) of the house-metro station dataset. In detail a 2-skyband query will return points H1, H6, H7, H8, H9 (which are dominated by at most 0 points) , H3, H11 (which are dominated by at most 1 points) and H2 (which is dominated by at most 2 points).

A naïve approach to process a k-skyband query is to perform an enclosure (window) query on the R-tree, for every point $p(p.d_1,p.d_2) \in DS$, to count the points that exist in the region $[0,p.d_1)[0,p.d_2)$. If there exist up to k points in this region then the point p belongs to the skyband. Since this approach is inefficient, because the number of enclosure (window) queries required is equal to the cardinality of the dataset, a more efficient approach involves the processing of k-skyband query with the BBS algorithm. As with the original skyline the algorithm maintains its progressiveness and its only difference is that an entry is rejected only if it is dominated by more than k discovered skyline points.

2.4.11. Summary

The user can apply a dynamic skyline query if in addition with the original skyline computation, wants to apply a space transformation from a (i.e) 3-dimensional space to a 2-dimensional space, (and vice versa) or to find the skyline set based on a given query point. A reverse skyline query can help the user to identify if a given query point is desirable and interesting based on an existing dataset that may represent his/her preferences. A spatial skyline query can be applied when the user wants to find the skyline according to multiple query points such as the case of deploying several police cars to respond to multiple incidents. The Group-by skyline can help the user to identify the interesting points based on some common attributes. i.e find the best hotels in each 5-star category. A thick skyline can help the user to retrieve not only the skyline points but also some additionally points that may be interested to know even if they are not truly-interesting points but only nearly-interesting points (are very close to a truly interesting point). A top-K skyline query can help the user to retrieve the interesting points of a dataset even if his/her preferences are biased. In this example he/she prefers cars with twice as low consumption even if its horsepower is tripled lowered. With a k-representative skyline the user can retrieve a representation of the

original skyline, which is consisted from a smaller number of points than the original skyline. This representation can be based on dominance or distance from other representatives, depending on the selected query type. This type of query can be useful if the user wants to retrieve a general view of the skyline fast, without retrieving the full skyline. With a ε-skyline the user can incorporate the idea of top-k, k-dominating, thick and the k-representative skyline with one algorithm. An enumerating query will help the user to retrieve the skyline points and additionally the number of points that each skyline point dominates while with a k-dominating query can retrieve the k-points that dominate the most points. Finally, a k-skyband query will let him to retrieve points based on the number of points that dominate a point which can be useful in cases where the user wants to know the dominance relations.

A performance analysis between BBS **[2, 3]** and NN **[37]** based on the application of the various queries types can be found in **[3].**

Query type	Specific Algorithms	Based-on	incorporates
Constrained Skyline **[2, 3]**	Modified BBS or NN	BBS or NN	MBRs
Dynamic Skyline **[2, 3]**	BBS	BBS	*mindist*, distance functions
Spatial Skyline **[46, 47]**	B^2S^2	BBS	Convex hull
	VS^2	-	Voronoi diagram / Delaunay graph
	VCS^2	VS^2	Voronoi diagram Delaunay graph
Reverse Skyline **[14]**	BBRS	BBS	Global skyline
	RSSA	-	Global skyline / Approximation of skyline
Group-by Skyline **[3]**	Modified BBS	BBS	Secondary R-tree / sorting
Top-k Skyline **[2, 3]**	Modified BBS	BBS	*mindist*, distance function
Thick skyline **[60]**	Sampling & Pruning	-	sampling / Strongly Dominating Relationship
	Indexing & Estimation	*Index* **[35]**	sorting
	Microcluster-based	-	microcluster-based index
K-representative **[62]**	Greedy	BBS	sort-merge paradigm
	FMGreedy	Greedy - BBS	FM-algorithm**[63]** / FM sketches
	R^{FM}-tree	BBS	R^{FM}-tree **[62]** / FM sketches
Distance-based K-representative **[65]**	2D-opt	-	R-tree / Covering circles
	I-greedy	-	R-tree / farthest neighbor search
ε-skyline **[68]**	ε-sfs	SFS	specific monotone function
	IFR	-	extra set ToExpand whith MBRs
Enumerating query **[2, 3]**	Modified BBS	BBS	R-tree, find dominance regions
K-dominating query **[2, 3]**	Modified BBS	BBS	Enumerating query, constrained skyline
K-skyband query **[3]**	Modified BBS	BBS	pruning restrictions

TABLE 15: SPECIFIC ALGORITHMS FOR EACH QUERY TYPE.

Table 15 outlines the basic algorithms developed for the various types of queries mentioned and notes the fundamental skyline algorithm that is based (if applicable) and the specific structures (geometric) or techniques (approximation) that may incorporate.

Type	Method	Size of resulted set
Constraint skyline	Region restrictions	$K \leq S$
Dynamic Skyline	Space transformation	S
Spatial Skyline	Geometric structures	S
Reverse Skyline	Space transformation	S
Group-by skyline	Grouping attributes	S
Thick skyline	Approximate selection	$K \geq S$
Top-K (ranked)	Point-wise ranking	Exactly k points, $\emptyset \subset K \subseteq Ds$
K-representative	Exclusive domination	Exactly k points, $K < S$

Distance-based K-representative	Distance aware	Exactly k points, K < S
ε-skyline	Multiple methods	∅⊆ K ⊆Ds
K-skyband query	Domination	K ≥ S
Enumerating query	Domination	S
k-dominating query	Exclusive Domination	Exactly k points

TABLE 16: SKYLINE QUERIES APPROACHES

Table 16 illustrates the various skyline related approaches, the general method that is used to retrieve the skyline points of a dataset Ds and the size k of the resulted skyline set in a general case, compared to the size S of conventional skyline query.

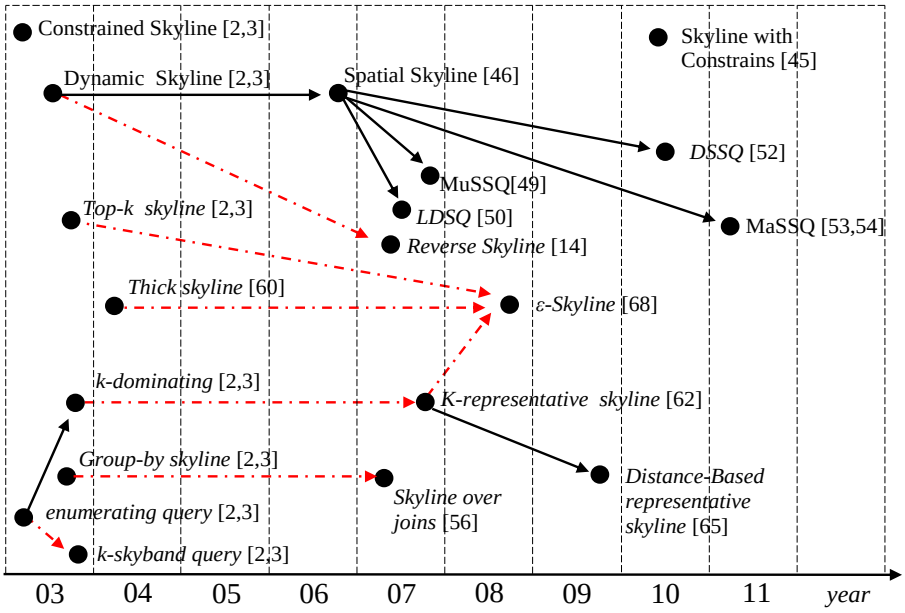


FIGURE 31: CHRONOLOGICAL ORDER OF BASIC SKYLINE QUERIES.

Figure 31 illustrates the various queries. The black lines indicate that the algorithm heavily depends or improves a previous algorithm and the red dashed line indicates that the algorithm shares some general main ideas to compute the skyline.

2.5. Applications

Through several years of research skyline queries where applied in many applications and data specific environments. This section primarily outlines the applications of skyline queries that are related to this Thesis and then summarizes the work on the numerous research topics that exist.

2.5.1. Skyline Queries Over Temporal Data

The increasing interest in maintaining numerous time-varying data versions and in supporting queries and trends analysis for decision making using these data led to the publication of over 2,000 research papers, to a comprehensive glossary of terminology [70], surveys and books in temporal databases. These usually refer to two types of time, valid time and transaction time. The first corresponds to the time when a fact is true in the real world. The second is the time during which a piece of data is stored in the database. These time intervals might be different for the same data tuple. Databases that combine both these types of time are called bi-temporal.

Surveys of access methods for efficient query processing over temporal data are found in [71] and [72]. A large number of these methods are modifications of the traditional B^+-tree access structure, such as the Multi-Version B-tree [73] and the Overlapping B^+-trees [74]. They usually index tuples in the form $<k, t_1, t_2>$ in which k is the key to a database relation and $[t_1, t_2]$ is a time interval, which in most of these cases is the transaction time. Another group of methods employs mapping strategies and transformations such as the mapping of time intervals to single-

dimensional points in MAP21 [75] or the interval transformation in the Interval Space Transformation method [76]. These methods usually index valid time ranges of the form [v_1, v_2]. Another cluster are extensions of space partitioning indexing structures such as the Segment R-tree [77], 4R-tree [78], 3D R-tree [79], MV3R-tree [80], the RI-tree [81], etc. Most of these methods can efficiently support tuples of the form <k, t_1, t_2, v_1, v_2>, and can index both temporal and bi-temporal data.

While several temporal queries, joins and semi-joins have been explored for several application domains, a query that has not been discussed yet in temporal databases is the skyline query and its variants. This Thesis will be addressing the problem and propose algorithms for computing efficiently the well-known static, dynamic and reverse skylines for temporal data. A closely related work is provided in [82], in which the interval skyline query is introduced for time series applications. The query returns the time series which are not dominated by any other time series in a time interval. However, the properties that hold in the time series environment are totally different than that in the field of general temporal (non time-series) data and therefore the proposed algorithm is not applicable in temporal and bi-temporal databases. Finally, the authors in [83] refer to the term *temporal skyline* which is also introduced in this thesis, however with a different meaning to support the so called convex skyline query for sets of spatiotemporal objects in privacy aware environments in which the disclosure of aggregated values of objects is only allowed.

2.5.2. Parallel and Big Data Skyline Computation

With vast amount of data available and the presence of fast networks, large clusters of commodity machines, multi-core processors and large amounts of shared memory many parallel programming frameworks were developed. Two of the most common programming frameworks used in skyline computation for parallel and distributed processing are the *MPI* (Message Passing Interface) [84] and *MapReduce* [85]. Each one of them fits best for particular problems [86] depending on whether there are data exchanges between nodes or not [87]. Another parallel programming framework commonly used is OpenMP (Open Multi-Processing) [88]. Each programming framework is designed to solve problems in different environments with *OpenMP* to fit best in shared memory systems, *MPI* in distributed memory systems and *MapReduce* in big data processing [86]. In the Era of AI, we cannot ignore parallel computing on general-purpose graphics processing units (*GPGPU*) that can be performed with *CUDA* [89] and *OpenCL* [90].

Taking into account the previous frameworks, a study on skyline computation in *OpenMP* was conducted in [91], where the authors parallelized the *BBS* [3] and *SFS* [41] algorithms and proposed the nested-loop-based *SSkyline* and the divide-and-conquer-based *PSkyline* algorithms for *skyline query* computation. In [92], the authors proposed the *Hybrid* and *Q-Flow* skyline algorithms, which use a data structure maintained in the shared memory managed by *OpenMP*. A pivot-based technique to partition the space in such a way to minimize the number of comparisons, taking into account the incomparability of points and partitions, was proposed in [93]. In a similar concept, [94] uses an angular-based space partitioning technique to solve the skyline computation problem. For *parallel skyline* computation, the authors of [95] used angle-based space partitioning based on hyperspherical coordinates. A similar approach [96] performed partitioning based on Hyperplane-projections. In [97], the authors proposed four algorithms based on load balanced grid-based partitioning techniques.

In a different environment, the authors of [98] implemented the *SFS* [41] and *SaLSa* [42] algorithms in *CUDA* [89] and compared them with the *GNL* algorithm [99], a nested-loop skyline algorithm for GPU, based on CPU-based *BNL* [8]. Another work for GPU-based skyline computation is the one in [100] where the authors proposed the *SkyAlign* algorithm along with a GPU-friendly, grid-based tree structure and compared them with the work on [98].

The first work that studies skyline queries on the MapReduce programming framework is [101] that ported the original *BNL* [8], *SFS* [41] and *Bitmap* [35] algorithms, in *MapReduce* with the *MR-BNL*, *MR-Bitmap* and *MR-SFS*. In [102] the authors used the *BNL* [8] algorithm with an angular partitioning approach to solve the skyline problem. Through their work, they proposed the *MR-Dim*, *MR-Grid* and *MR-Angle* algorithms. The idea of *minimal algorithms* in *MapReduce*,

which have in goal the load balancing among nodes and the minimization of space, CPU, I/O and network, was proposed in [103]. Among ranking, group-by and semi-join algorithms, the skyline queries were discussed developing the *Minimal-Sky* algorithm and compared it to the *MR-SFS* from [101]. The authors of [104] proposed the computation of histograms to initially prune the non-interesting points and consequently partition the dataset in a later phase. With this technique, they managed to compute the *skyline* and *reverse skyline* queries with the *SKY-MR* and *RSKY-MR* algorithm respectively. It is worth mentioning that the most relevant work to this is the algorithm *MR-BNL* [101]. Their work is also applicable to the *MPI* framework and to multicore environments such as in *OpenMP.* In [105], the skyline computation by the MR-GPSRS and MR-GPMRS algorithms was achieved with a bitstring representation on the original tuples and a grid-based partitioning approach which reminds of the approach followed by the *Bitmap* [35] and D&C [8] algorithms. The authors compared their algorithms with the *MR-BNL* [101] and *MR-Angle* [102]. The angle-based partitioning approach was also used in [106] with the *PGPS* algorithm which works similar to the *BNL* [8] and *SFS* [41]. Additional partition-aware filtering approaches were proposed and used in the *APF-PGPS* and *PPF-PGPS* algorithms to balance the skyline candidates over the partitions. Moreover, the authors compared their work with the algorithms in [102] and [105]. In [32] the authors use the spatial and geometric properties of the dataset to prune the dataset with the use of *SpatialHadoop*, an extension of *Hadoop*. Their proposed algorithms *SKY-FLT* and *SKY-FLT-SORT* for skyline computation efficiently maintain checkpoints to prune the rest of the dataset.

method	query type	novelty	Indexing / sorting / partitioning	Pruning techniques	No. Jobs	Skyline algorithm in use	Dimensionality / Cardinality	nodes / memory
[101]	Skyline	Porting	Horizontal	Inherited	2 jobs	BNL, SFS, Bitmap	10d /1B	8 / 4GB
[102]	Skyline	Angular partitioning	Vertical & Horizontal	Inherited	2 jobs	BNL	10d / 10TH	32/ 4 GB
[103]	Skyline & others	Minimizing costs	TeraSort	Inherited	2 jobs	BNL	2d/ 2.5B	56 / 4GB
[104]	Skyline & Reverse Skyline	Sampling, histograms	QuadTree	Partition aware with local filter points	2 jobs	-	10d / 4B	20 / 4GB
[105]	Skyline	Sampling, Grid partitioning with BitString representation	Grid with BitString mapping	Partition aware with local filter points	2 jobs	-	8d / 3M	13 / 28 GB (total)
[106]	Skyline	multiple partition-aware filtering mechanisms	Angle-based & Grid-based partitioning	several partition aware filtering approaches	2 jobs	-	5d / 13M	12 / 2 GB
[32]	Skyline	SpatialHadoop (sampling with two level index)	R-tree	checkpoints	1 job & (1-time job for indexing)	Similarities to BNL	2d/ 500M	17 / -
[107]	Skyline	Single job, sampling with two level index	TLG (Two-level Grid) index	Partition and intra-partition filtering	1 job	-	4d / 10M	8 / 16 GB
[108]	Skyline & Reverse Skyline	Reverse Skyline in SpatialHadoop, high cardinality datasets	R-tree	Partition and intra-partition filtering	1 job for Skyline, 2 jobs for Reverse Skyline & (1-time job for indexing)	Similarities to BNL	2d / 2.4B	6 / 16 GB

TABLE 17: MAPREDUCE-BASED SKYLINE QUERY COMPUTATION APPROACHES.

Finally, Table 17 summarizes the work on skyline query processing related to *MapReduce*. The table organizes the work in chronological order and highlights the novelty, the approach followed and some basic technical aspects of each work. As indicated, the trend followed in the early stages of *MapReduce* was to port the existing sort-based algorithms for skyline computation. The research evolved by proposing sampling techniques to build indexes or identify points with high pruning power in multiple jobs. The use of indexes with additional pruning mechanisms seems to be the *defacto* approach nowadays in which two level indexes appear with techniques that try to solve the problem in one *MapReduce* job. To the best of our knowledge, the only work that studies

reverse skyline queries [14] in *MapReduce* is the one in [104]. Moreover, the only work that studies skyline queries in *SpatialHadoop* is the one in [32]. The only work that retrieves the skyline set in a single job is [107], the [32] and the work presented in this Thesis [108] which needs a one-time job to perform the indexing of dataset in contrast to other works that re-index the dataset in every execution of the algorithm. Additionally, it is the only one that studies reverse skyline queries in *SpatialHadoop* and additionally shows how the skyline queries on *SpatialHadoop* performs in a high cardinality dataset of 2.4B points.

2.5.3. Data mining

The work of skyline queries related with data mining approaches is quite limited. In most cases researchers use clustering approaches to estimate the skyline or use the skyline operator to select the best candidates or undominated solutions.

More specifically, clustering approaches are employed in [109] where authors use k-means clustering to identify k-representative skyline and in [110] where a multi-objective genetic algorithm-based clustering approach is used to identify the pareto-optimal front and find the skyline. The notion of dominance is used in [111] where authors use the skyline operator to identify a set of approximate undominated graph clustering solutions and in [112] to identify un-dominated subgraphs. In authors [113] use the skyline operator as a filtering approach in a classification task over biological data. In terms of identifying the best candidate solutions authors of [114] try to identify the most suitable classifier in terms of accuracy, detection rate and false alarm rate using the skyline operator. Finally, in the work [115, 116] presented in this Thesis the skyline operator is used to estimate the decision boundaries in a classification process.

2.5.4. Other Applications

This section will present the rest numerous research topics on skyline queries. This includes the skyline computation of skyline queries in a portion of the original dataspace, distributed skyline computation, applications of skyline queries in specific data environments, continues skyline computation in streaming environments, security related skyline applications, the use of skyline queries to maintain the Quality of Services, the computation of skyline queries in spaces different from the Euclidean such as metric space, the cardinality estimation of the answer of a skyline query and the efficient update and maintenance of skyline queries when the original dataset changes.

2.5.4.1. Subspace and Space Partitioning

The fundamental methods for skyline computation are optimized and rely on the fact that the dimensionality of queries is fixed and concerns the full space of the dataset (take into account all the dimensions/attributes of the dataset). Nevertheless, different users may be interested about different dimensions/attributes of data and therefore may want to retrieve the skyline by comparing only a specific subset of all dimensions/attributes. Additionally, a full space skyline query in high dimensional space may return too many interesting points to the user which will not allow him/her to make an appropriate decision. This problem reveals a different scenario in which a query is placed over fewer dimensions than those of the full space. Formally, given a set of d-dimensional points, a skyline query can be issued on any subset of the *d* dimensions. This subset will be called *subspace* and the corresponding skyline query on those dimensions subspace *skyline query*.

MULTIPLE SUBSPACE COMPUTATION

Multidimensional subspace skyline computation was proposed simultaneously by two different groups of authors in [20] and [117]. The methodology that they follow is different but the main idea remains the same and is to compute the skylines of all possible subspaces forming a lattice structure similar to the data cube [118, 119]. The authors of both groups combined, extended and improved their works in [120]. The initial problem that they state is that none of the existing methods considered skyline computation in subspaces. Authors in [121, 122] proposed the *Compressed SkyCube (CSC)* which represents the *complete SkyCube* preserving the essential information of subspace skylines without accessing the whole dataset. The initial reason for this work was that the previously described method SkyCube (or complete SkyCube) did not take into account that the dataset is not always *static* but rather could be *dynamic* (not to be confused with

the *dynamic skyline)*. Authors in **[117]** discussed primarily the semantics of subspace skyline queries and the importance of the dominance relationships in the subspaces. They studied the *skyline membership query*, which tackles "why and in which subspaces an object belongs to the skyline", by introducing the notions of *skyline group* and using the general idea of *decisive subspaces*. In **[123]** authors improved their work on **[117]** in order to sufficiently address the efficient skyline group and decisive subspaces computation. In this work also developed the *Stellar* algorithm which computes *skyline groups* and *decisive subspaces* without searching all subspaces for skyline points, by exploiting the skyline groups formed by the full space skyline points.

SINGLE SUBSPACE COMPUTATION

Previously proposed methods, related with the subspace skyline computation computed the skylines of *all* the possible subspaces. This approach was selected because it is not known (unpredictable) in which and how many dimensions a user may want to retrieve the subspace skyline, so it is needed to compute every possible subspace skyline. From another perspective many times most of the users perform queries that are related with a small subset of dimensions (and might also be the same in their majority) with respect the full space. Differentiating, authors in **[124]** studied the computation of the skyline of *one* specified subspace, as opposed to all. In **[125]** extended their work where they discuss about the applicability of existing full-space skyline algorithms in subspace and extend the *SUBSKY* algorithm to compute *k-skyband* **[3]** and *top-k* queries **[2, 3]** in subspace.

TOP-K AND K-DOMINANT

To deal with the problem of returning to many interesting points in high dimensional spaces, when different subspaces are considered, authors in **[126]** focused on ranking skyline objects and introduced a new metric called skyline *frequency.* This metric ranks the interestingness of a skyline point and retrieves the skyline points in a top-k fashion. Moreover, as dimensionality increases the chance of one point to dominate another is very low. This leads in the retrieval of numerous skyline points, which cancels any interesting insights on the dataset. An efficient approach is to relax the notion of dominance to k-dominance in order to consider only *k* among *d* dimensions and retrieve only important and meaningful skyline points. For that reason, same authors of previous work (**[126]**) proposed in **[127]** the k-dominant skyline query (not to be confused with k-dominating queries). A k-dominant skyline query retrieves a representative subset of skyline points from a high d-dimensional dataset.

SPACE PARTITIONING

The access order of data points has direct impact on the performance of algorithms since the early identifications of dominant and highly-dominating points can eliminate unnecessary domination comparisons. Additionally, pairwise point-to-point dominance comparisons have considerably computational and time cost which can be avoided by block based-comparisons. Taking into account the previous considerations authors in **[128]** proposed an approach based on the popular dimension reduction technique, Z-order space filling curve (or Z-order curve, in short) **[129, 130]** which carry many good geometric properties for skyline processing. In **[131]** authors extended their work, proposed new algorithms and studied the problem of ranking and subspace skyline query processing. Authors in **[132]** proposed the Lattice Skyline (LS), to answer skyline queries of low-cardinality domains, which uses a static lattice structure to determine the dominance between the various combinations of distinct attribute values in the dataset.

INCORPORATING INCOMPARABILITY

While most of the methods are dominance based, incomparability is very useful in high-dimensional spaces since most pair of points become incomparable. To minimize the computational cost in high dimensional spaces authors in **[133]** proposed a progressive object-based space partitioning (OSP) algorithm, which recursively divides the d-dimensional space into 2^d separate partitions taking into account the incomparability property. Authors in **[134]** proposed the BSkyTree which tries to find a cost-optimal strategy for skyline processing by exploiting the properties of both dominance and incomparability.

In multiple subspace skyline computation the algorithms SkyCube [20] and Skyey [117] where parallel developed by a different group of authors, reasoning about the same problem. Continuing, their works where individually extended. From previous described algorithms, partition-based approaches are the algorithms Z-search, LS, OSP and BskyTree. Partitioned based approaches that are based on incomparability are OSP and BSkyTree. From these approaches Z-search can be extend on subspace, k-dominant and k-skyband. The OSP can be extended to k-dominant skyline. Z-search and LS are based on dominance comparisons while OSP and BSkyTree on incomparability.

The **Table 18** summarizes the state-of-the-art algorithms related with subspace and space partitioned skyline computation. The column *Approach* corresponds to whether the algorithm pre-computes the results, uses indexes or sorting and counting techniques to answer a requested query. The *No. of subspaces* corresponds to whether the algorithms make their computations for all the possible subspaces, a single subspace or is generally applicable to full-space computation as with the partitioned based approaches. The column *Dataset* corresponds to weather the algorithm is applicable in dynamic datasets where updates and deletions occur.

Algorithm	Approach	No. of Subspaces	Dataset	incorporates
SkyCube [20]	pre-materializing	all	static	cost sharing strategies
Compresed SkyCube [121, 122]	pre-materializing	all	dynamic	minimum subspaces
Skyey [117]	pre-materializing	all	static	Skyline groups – decisive subspace
Stellar [123]	pre-materializing	all	static	skyline groups – decisive subspace
Subsky [124, 125]	index	single	dynamic	$L\infty$ distance, anchor points
Skyline frequency [126]	counting	single	static	maximal dominating subspace
k-dominant [127]	counting/ sorting	single	static	conventional skyline - nested loops
Space Partitioning				
Z-search / Z-sky [128, 131]	sorting - index	full space/single	dynamic	Z-order curve, ZBtree
OSP [133]	Index	full space	dynamic	incomparability, LCRS tree, bitmaps
LS [132]	Lattice structure exploration	full space	static	lattice structure
BSkyTree [134]	Cost-based partitioning	full space	static	pivot points, incomparability

TABLE 18: STATE OF THE ART SUBSPACE SKYLINE ALGORITHMS.

2.5.4.2. Distributed Skyline Computation

Due to the high skyline query processing cost of centralized architectures**,** research has focused in distributed skyline query processing. However, previously proposed algorithms cannot be directly applied in distributed environments and thus specialized approached where proposed.[1]

VERTICAL PARTITIONING

An early proposed work on distributed skyline computation considers the vertical partitioning of the dataset. This partitioning approach was not adopted by the rest of the researchers due to the wide adoption of horizontal partitioning on highly distributed environments, such as peer-to-peer networks. More specifically, the work in [135] was the first that studied the problem of skyline query processing in distributed environments and especially in a specialized Web setting where each one of the dimensions are stored on a different Web-accessible site (source/database). Authors in [136] improved the previous work by proposing the *PDS* (progressive distributed skylining) algorithm. Another work related with vertical data partition is the one in [137] where authors assume that the dataset is vertical partitioned in a number of arbitrary servers and each server stores an arbitrary number of dimensions.

[1]The work of [469] is the only survey in skyline query processing, which is focused on parallel distributed skyline query processing.

HORIZONTAL DATA PARTITIONING

The following methods concern horizontal data partitioning, in which a portion of the dataset is stored by a peer without taking into account the partitioning imposed on the dataspace. This defers from the approach of horizontal dataspace partitioning where a partitioning scheme is applied to the dataspace and the points of each partition are assigned to peers. In these methods peers (servers) can communicate with their neighboring peers, through a coordinator, or through the use of a backbone structure. Thus, this approach does not consider any kind of overlay structure, in which a logical network is built on top of a physical one without considering its physical network structure. More specifically, the work on **[138]** is focused in distributed skyline computation on mobile devices that communicate without routing information in shared-nothing environments and especially over ad-hoc networks (MANETs). In this scenario data are stored in a number of light-weighted mobile devices where each device is able to communicate only with its neighbors (devices that are in its communication range) by exchanging messages. The communication between all devices can be achieved via multi-hops. Each mobile device stores a portion of the whole dataset (that can be overlapping) which corresponds to a portion of the data that are related with the geographic area that it covers. The partitioning placed in this case concerns horizontal data partitioning. On **[139]** authors study the problem of subspace skyline query processing in super-peer networks (large scale P2P networks) and proposed the SKYPEER framework where the dataset is horizontally distributed across peers. In this type of networks there exist a number of super-peers among the ordinary peers which have special abilities due to their enhanced features. Super-peers are linked through a backbone and peers are connected to super-peers. Each super-peer maintains information about the peers that have assigned on him to achieving efficient routing.

HORIZONTAL DATA PARTITIONING WITHOUT OVERLAY NETWORKS

The works that follow are based on horizontal data partitioning but do not consider any underling overlay network. That is the query originator can communicate with all the existing peers to compute the skyline set. Based on this, in **[140, 141]** authors proposed the filter-based PaDSkyline algorithm for parallel constrained skyline query processing in which they assume that no overlay exists and any query originator can directly communicate with all servers. This approach employee and extends the single-point filtering method of MANET **[138]** with the difference that uses multiple filtering points rather than just one, considering that wired connections are faster and more reliable than the wireless. In **[142]** authors proposed a progressive feedback-based distributed skyline (FDS) algorithm which assumes that a small number of servers are geographically distributed and connected through the internet. Data are partitioned horizontally and assume no overlay structure. FDS is focused on minimizing the transferred data among the network.

HORIZONTAL SPACE PARTITIONING

The following works reasons about distributed skyline computation assuming the incorporations of a dataspace partitioning technique. Thus, each node will be responsible for a disjoint partition of the data space. This scenario differs from this in the previous since the partitioning is imposed to the dataspace rather to the dataset itself. The methods that will be described can by categorize into two classes. DHT – based (distributed hash table-based), such as CAN **[143]** and balanced-tree based such as BATON **[144]**. On this scope, in **[145]** proposed the progressive and parallel distributed skyline algorithm (DSL). The algorithm deals with constrained skyline queries, which return the skyline set of a given region, on a shared-nothing architecture. DSL is based on grid-based data space partitioning techniques which horizontally partition the space. The data partitioning is determined by the structured P2P overlay networks CAN **[143]**. CAN is a distributed, decentralized P2P infrastructure, based on a logical d-dimensional Cartesian coordinate space, which incorporates a distributed hash table (DHT) for point and server multi-dimensional indexing. Authors of **[146]** proposed the Skyline Space Partitioning (SSP) approach which is based on BATTON **[144]**. BATON instead of using a distributed hash table (DHT) as CAN **[143]** uses a distributed balanced tree for indexing of nodes. As opposed with the previous method SSP processes the unconstrained skyline queries which retrieves skyline points from the whole space. Authors in **[147]** extended their work on **[146]** and proposed the SkyFrame framework which can be applicable on BATON and other structured P2P overlay networks. In **[148, 149]** authors proposed the iSky algorithm which is similar to the SSP/skyframe **[146, 147]** and is based on the

BATON overlay network [144]. The differences from the SSP/SkyFrame, is the use of the iMinMax [150, 151] data transformation, as similarly used in the Index algorithm (section 2.3.4).

ANGLE-BASED PARTITIONING

Previously proposed methods for distributed skyline computation widely adopt grid-base space partition techniques. In [95] authors proposed an angle-based space partitioning approach which uses hyper-spherical coordinates [152] of points in order to partition the space in such a way to increase the efficiency of parallel distributed skyline query processing.

Algorithm	Incorporates	Routing	Topology	Skyline Query Type	Main drawbacks
BDS-IDS / PDS [135]	Vertical partitioning – sorting	Direct / initiator to peer	Web		Vertical partitioning
MANET [138]	Filtering point	Breadth/depth first	Manet	Subspace/ Constrained/ Dynamic	Exhaustive routing
SKYPEER/ SKYPEER+ [139]	Sorting / threshold value Ext-skyline	Super-peers	Super-peers	Subspace	Existence of super-peers
PaDSkyline [140, 141]	Multiple filtering points & MBRs	Cluster-heads	Clusters	Subspace/ Constrained/ Dynamic	Flooding / Heavy load on cluster-heads
FDS [142]	Multiple-round filtering /sorting	Direct / initiator to peer	Web	Subspace/ Constrained/ Dynamic	Many rounds on large networks
DSL [145]	Partial ordering	Local routing table / neighbors	CAN	Constrained	Load balance – High cost on updates
SSP [146]	Filtering points/Partition ordering (z-order)	Balanced tree adjacent nodes	BATON	Constrained	Load balance
SKYFRAME [147]	As SSP + Border regions	Balanced tree adjacent nodes	BATON-CAN	Constrained	Load balance
Isky [148, 149]	Sorting/ filtering points/ threshold value /	Balanced tree range search	BATON		Load balance
Angle-based partitioning [95]	Hyper-spherical coordinates	-	-	-	Issues on its application

TABLE 19: FUNDAMENTAL ALGORITHMS ON PARALLEL AND DISTRIBUTED SKYLINE COMPUTATION.

The Table 19 outlines the fundamental key aspects of the previous proposed methods.

2.5.4.3. Attribute & Data-Specific Applications

Previous methods considered that all attributes of all dimensions are available, for all points. Additionally, there is the case of incomplete datasets, where the points miss some of their dimensions/attribute values, partial order datasets, where the ordering of attributes can't be defined or is defined differently by each user and finally uncertain datasets where an object may have different instances that can occur with different probabilities.

PARTIALLY ORDERED DATASPACE

The previous studies focused on total order (TO) domains (dataspace). That is datasets where their attributes have an internal ordering such as numbers. In these domains it is easy to understand which attributes are preferable than others. The lack of ordering or preference among a pair of attributes indicates that a domain is partially ordered (PO). Authors in [24] focused on skyline computation over partial-ordered domains. This type of domains among others can include intervals, hierarchies, domains of set values and preferences. In [153] authors extended the work of [24] and presented a progressive framework named Topologically-sorted Skyline (TSS). The work in [154, 155] considers the case where different users have different preferences and thus the ordering imposed on the dataset changes for each user.

INCOMPLETE DATA

Another data-related skyline computation approach is the one in [25] which assumes that data are incomplete, meaning that have missing values in some of their dimensions/attributes. Most of the algorithms assume data completeness on all dimensions and transitivity in the dominance relation. However, this is not always the case. On incomplete data, the transitivity does not always hold. The closest work to this is the one in [127], which also does not assume that transitivity holds. In [156] authors studied the skyline queries over crowd-enabled databases. Crowd-enabled databases deal with incomplete data during runtime by requesting missing values or complete tuples from other sources.

UNCERTAIN DATA

The problem of skyline computation on uncertain data was tackled in [26, 27] where authors proposed a probabilistic skyline model and additionally the p-skyline. Some conditions that may impose uncertainty on data are limitations on receiving and measuring data, missed or delayed data reports and randomness of data. In general, the probability of an object to be in the skyline, is the probability that the object is not dominated by another object. Moreover, in [157] authors propose efficient methods to compute the skyline probabilities on all objects. In their work proposed a more general uncertain model where the instances of each uncertain object may have different probabilities to occur and that the probabilities of all instances may sum up to less than 1, meaning that may exist an instance of an object that is not known to us. In [158] authors study the same problem with [157] and propose an asymptotically faster algorithm for the worst-case. The algorithm uses the same partitioning technique as in [157] but handles the partitioned sets more efficiently. In [159] authors further studied the problem of [157]. In their work compute the exact skyline probabilities of all objects in high-dimensional datasets by incorporating a ZB-tree [128]. In [160], authors reason about reverse skyline computation over uncertain data in monochromatic and bichromatic cases. In the monochromatic case the point (object) of interest and the query point (object) are of the same type and thus from the same dataset, while in the bichromatic case there exist two different types of points (objects). In [161] authors extended their work on [160] and proposed the probabilistic reverse furthest skyline (PRFS) which considers the case where minimization of preferences is desired rather than the maximization. Authors in [162, 163] reason about distributed skyline computation over uncertain data. Their scenario is based on the existence of a number of distributed sites that each one of them contains a number of uncertain data and a centralize server that processes the query. In [164], authors reason about contextual skylines taking into account the uncertainty in user's preferences rather the uncertainty of attribute values. Authors in [165] further studied the problem of [164] without taking into account the independent object dominance assumption that was considered in [164] and in which the object's dominances are considered as mutually independent events. Authors in [166] reason about the top-k skyline query computation on uncertain data.

TRADE-OFF & STOCHASTIC SKYLINE

The Trade-off skylines were first proposed in [167]. A trade-off is defined as, how much is willing to give from one dimension to gain an improvement to another dimension/attribute. This concept is primary reflected by the top-k retrieval paradigm using weighting over each attribute. This constructs a scoring (or utility) function which is used to compute the overall ranking of an object based on all of its attributes.

Finally, Stochastic Skyline queries where proposed in [168, 169]. In a stochastic model, the subsequent state of the system can be determined probabilistically based on previous states. As an example, a future stock price will be equal to the current stock price plus an unknown change that can be determined probabilistically. The proposed model uses, for each user, one scoring function for every dimension.

2.5.4.4. Continuous Skyline Computation

The existing skyline algorithms are designed to compute skylines over static datasets rather than dynamic, that occurs in streaming environments. Dynamic streaming data can be retrieved with the use of data-streams [170] which is a continuous stream of received data point.

In [16] authors reason about on-line computation in the presence of rapid updates of data such as in data-streams. Particularly the scenario concerns append-only data-streams [170] where there is not any deletion of the data elements till they expire and the elements are positioned and labeled according to their relative arrival ordering. Such type of streams are those of wireless sensors networks, where the data collected prior a specific time interval are discarded because they are not representative in comparison with the existing readings of sensors. Authors in [171] also proposed the skyline computation in data stream environments. In this scenario, they take into account only the tuples that arrive in a sliding time window [170]. That is the W most recent timestamps, where W is a parameter that defines the length of the window. The work in [172] studied the problem of continuous skyline computation on datastreams where the validity and expiration of points is determined with the use of time intervals. Each point received is associated with an arrival and an expiration time which essentially defines the time interval that the point will be valid. In [82] authors studied the problem of skyline query computation on time series [173]. Time series are useful since they can give information about events that happen in a specific time interval. As an example, these events could include the upload bandwidth consumption or the visiting rate of a web page, along the day, moth, year or any other time interval. The Table 20 outlines the fundamental key aspects of the previous proposed methods.

Algorithm	Incorporates	Environment	Indexing method	Skyline on # points
[16]	Stabbing queries [16]	Sliding-windows	R-tree	n-of-N & (n1,n2)-of-N
[171]	Dominance & anti-dominance regions	Sliding-windows	R-tree	N most recent
[172]	continuous time-interval skyline	Time-intervals	R-tree / Quad-tree	All valid received points
[82]	interval skyline	Time series	Radix priority tree	#Time-series* #timestamps

TABLE 20: FUNDAMENTAL ALGORITHMS ON CONTINUOUS SKYLINE RETRIEVAL.

2.5.4.5. Route Skylines Queries and Road Networks

Following will be discussed the in-route skyline algorithms that are related with the identification of efficient routes or detours on road networks and efficient locations that satisfy the desired minimization criteria among several user-defined points.

In [17] authors reason about skyline computation on road networks and particularly in-route skyline queries and in-route k^{th}-order skyline queries which concern normal domination and the points that are dominated by less than k other points respectively. The work on [18] consider route skylines in road networks with multiple preferences as opposed with the previous method. Authors of [49] reasoned about multi-source skyline queries where multiple query points are considered at the same time, in constrained space and especially on road networks. The multiple query points represent the locations or the points of interest from which a user wants to minimize its distance. The Table 21 outlines the fundamental key aspects of the previous proposed methods.

Algorithm	Skyline of	Data model	Incorporates	Attributes	Query points
[17]	detours	graph	NN-search / network distance	Single (length)	single
[49]	distance from data points	graph	NN-search / Euclidian-network distance & Dynamic/spatial skyline [2, Error! Reference source not found.]/ [46, 47]	Single (length)	Multiple
[18]	preference-based routes	multiple-attribute graph	Pareto optimality	Multiple attributes	start / destination

TABLE 21: FUNDAMENTAL ALGORITHM IN IN-ROUTE AND ROAD NETWORK SKYLINE COMPUTATION.

2.5.4.6. Security

The works that follow outline security related approaches that are based or use skyline queries and more particularly location-based skyline queries, user's privacy on anonymized datasets and queries over encrypted data.

Authors in **[22]** study the problem of authentication of location-based skyline queries. The scenario that is followed assumes that the spatial data are stored in a spatial database and are outsourced to a location-based service provider (LBS) which will handle the queries issued by users. In **[174]** authors proposed an additional method to generate VOs for spatial skyline queries. Their goal is to reduce the communication cost by reducing the number of digests to be reported and thus the size of the VO that is sent to the user. The work on **[175]** reasons about privacy in the presence of external knowledge. Their work builds upon and extends the work of **[176]** and quantifies the adversary's external knowledge in order to identify privacy threats and enforce privacy requirements. Authors in **[177]** reason about skyline queries over encrypted data. Their approach uses the SFS algorithm **[41]** in order to identify the skyline and a set of invertible matrices as the key of their encryption scheme.

2.5.4.7. Quality of (web) Services

The identification of the best web service among several similar is a multi-criterion decision problem since the optimization of various criteria is needed. In **[19]** authors study the problem of efficient selection of web services.

In **[178]** authors additionally considered the problem where the quality of the various services and service providers change over time. In many cases the aggregate QoS values may not perfectly reflect the actual performance of a service that is given by a service provider. Additionally, a service provider may not deliver the services according to the quality that declares. In **[179]** authors reason about the cloud-based web service composition. In their work, they use the skyline operator to prune unqualified services and reduce the related search space. In the next step they perform a Particle Swarm Optimization (PSO) **[180]** in order to find the optimal services based on the user's QoS constrains.

2.5.4.8. Metric Space

Skyline computation in metric space **[181]** was first proposed in **[23]**. In this work the dataset belongs to a metric space rather than a multi-dimensional space (i.e. Euclidean) as opposed in previous works. The reason that metric spaces involved to the skyline computation is because in many cases the dataset cannot be represented as vectors, something that is fundamental for the *Euclidean space*. An example in bioinformatics is the *DNA searching problem* where the *DNA sequences* are usually modeled and represented by strings and is desired to find the strongest sequence similarity.

In **[23]** authors proposed a *triangle-based pruning* method that incorporates the *triangle inequality property* in order to safely and efficiently prune the dataset (since distance computation can be very expensive **[182]** in metric spaces). Additionally they proposed an efficient Metric Skyline Query (*MSQ*) procedure that incorporates the M-tree **[183]** metric index structure, in order to retrieve the metric skyline points without scanning the whole dataset and without making any particular assumption about the data format and the metric distance function. In **[184]** authors extended their work on **[23]** by constructing an optimized metric index structure in order to minimize the cost of the metric skyline retrieval. In **[185]** authors try to improve **[23]** by proposing the *dynamic indexing* and the *k-dispersion* techniques in order to minimize the number of computations.

2.5.4.9. Cardinality Estimation

In general skyline cardinality computation tackles the problem of "curse of dimensionality" of skyline computation in high dimensional spaces. The estimation of the cardinality of the dataset can help to perform specific queries to reduce the size of the skyline set that is returned. Techniques that try to reduce the number of skyline points that are returned are the *skyline frequency* and *k-dominant* **(section 2.5.4.1).**

The cardinality of skyline queries was studied in [186] which proves that the skyline cardinality can be defined as $\Theta((\ln n)^{d-1}/(d-1)!)$, where n is the cardinality of the dataset and d its dimensionality. This indicates that the skyline cardinality increases with the dimensionality. Specifically, in [186] authors try to estimate the cardinality of the skyline query results based on the initial dataset, without any other assumption. In [187] authors extend the work of [186] in order to handle numerical and categorical attributes and additionally different distributions by proposing a Log Sampling (LS) approach. The drawback of the previous proposed algorithm is that is based in a hypothetical empirical model. Authors of [188] extended the work on [187] and proposed the kernel-based (KB) skyline cardinality estimation approach which is heavily based on kernels [189]. The drawback that the KB approach has is that it needs to perform complex computations. Additionally, the integration of PDF function over IDR regions suffers from the curse of dimensionality. For that reason, authors in [190] proposed the purely sampling-based (PS) approach and compared their method with LS and KS method. The Table 22 outlines the fundamental key aspects of the previous proposed methods.

Algorithm	Estimation	Dataset applicability	General approach
Log Sampling (LS) [187]	$\lvert SKY_{ds}\rvert = A(\log(n)^B)$	Not on clustered	Two samples
Kernel-based (KB) [188]	$\lvert SKY_{ds}\rvert = \lvert ds\rvert \times \frac{1}{\lvert s\rvert}\sum_{p\in SKY_s}(1-\Omega_p)^{\lvert ds\rvert-\lvert s\rvert}$	ANY	Samples based on Kernels
PS [190]	$\lvert SKY_{ds}\rvert = \left(\frac{\lvert SKY_s\rvert}{m}\right)\times n$	ANY	Single sample

TABLE 22: FUNDAMENTAL ALGORITHMS IN SKYLINE CARDINALITY ESTIMATION.

2.5.4.10.Skyline Updates & Maintenance

Researchers focused on skyline maintenance in order to efficient maintain the skyline when updates or deletions occur on the dataset. The reason of this research interest is that the update of an already computed skyline will have less computation cost than the recomputation of the skyline from scratch. This section does not assume the existence of data-streams or time series but rather considers that the updates (insertions or deletions) of points are placed over the existing stored dataset. From this perspective, authors in [3] where the first that studied the problem of incremental skyline maintenance when updates occur over the stored dataset. Their approach is named BBS-update. This approach was proposed by the authors that proposed the BBS algorithm and essentially discuss how BBS algorithm can efficiently maintain the skyline when various insertions or deletions occur. In [191] the Deltasky algorithm proposed that extends the BBS-Update [3] and reasons only about deletions since the insertion process was sufficient studied by the author that proposed BBS-Update. The drawback of DeltaSky is that it must scan all the sorted lists. In addition, if the skyline is issued on a high dimensional dataset the sorted lists will be large. ZUpdate and ZInsert + *ZDelete* [128, 131] efficiently update the skyline results if insertions or deletions occur by utilizing the sorting property of Z-order curve.

3. ONLINE SOURCES OF GEOSPATIAL DATA

In order to produce useful insights and new knowledge we should understand the data that we have and their nature. In order to build an information system for a specific sector an extensive study should be conducted about the available data, their quality and their integrability. This section examines geospatial free-of-charge data sources and discusses the various classes of available data. Those data can be used in maritime information systems which are innovative geographic information systems for study, monitoring and action-taking in maritime areas. In addition, this study will assist in identifying open research topics in relation to query processing, big data management or applied machine learning by understanding which are the main tasks that are performed over data.

3.1. Introduction

A maritime information system is a geographic information system (GIS) designed to capture, store, integrate, manipulate, analyze, manage, and visualize all classes of maritime geospatial data, capabilities which are serving a cross-section of disciplines. An increasingly cost-effective active maritime information systems market has also been developed, benefiting from an ongoing process of improvements in the hardware and software components of GIS. A variety of fields have benefited from the application of maritime information systems, made possible by this technological boost, from science, research, education, government, business, and industry, to domains such as public health, homeland security, natural resources management, astronomy, meteorology, climatology, naval archaeology, shipping transportation and logistics, etc.

Following are defined the classes of data which constitute valuable resources towards the development, performance tuning and efficient operation of maritime information systems, and subsequently surveys both the open and restricted data sources which provide free-of-charge these classes of real-world geospatial data. This is the first comprehensive study that classifies and analyses a spectrum of official online resources this wide, providing a thesaurus of high-precision real-word geospatial data to serve the needs of scientific research and development or educational work in the maritime information systems domain for purposes such as operational or benchmarking and experimentation or pattern recognition and data mining.

3.2. Examples of Historical & Modern Maritime Information Systems

The evolution and increasing importance of maritime technology has led to the current development of numerous powerful maritime information systems and projects at national and international levels. A number of examples include the Pattern Mining and Monitoring Ocean Eddies project **[192]**; the Long Range Identification and (Ships') Tracking system **[193]**, the SafeSeaNet **[194]**, and the ClearSeaNet project **[195]**, all three of the European Maritime Safety Agency **[196]**; the European Border Surveillance System **[197]**; the Maritime Surveillance project **[198]** of the European Defence Agency **[199]**; the CISE (Common Information Sharing of the Environment for the surveillance of the European Union Maritime Domain) initiative **[200]** and the Copernicus (European Space Agency's Global Monitoring for Environment and Security) program **[201]**, both of the European Commission's Directorate-General for Maritime Affairs and Fisheries **[202]**, together with the Copernicus' supporting projects DOLPHIN (Development of Pre-operational Services for Highly Innovative Maritime Surveillance Capabilities), NEREIDS (New Service Capabilities for Integrated and Advanced Maritime Surveillance) and SIMTISYS (Simulator for Moving Target Indicator System) **[203]** and the project related to the Copernicus, MyOcean2 **[204]**, **[205]**. A non-negligible number of such systems relate to specific seas, such as the Baltic Sea, which figures on the International Maritime Organization (IMO) **[206]** list for Particularly Sensitive Sea Areas **[207]**, and for which a large number of related programs has been developed, such as the Monalisa 2.0 **[208]**, the EfficienSea **[209]**, the BaSSy **[210]**, the SafetyAtSea **[211]**, *etc*.

3.3. Setting Out the Problem and Applying the Solution

Before discussing the various marine geospatial data classes and sources provided online, we will briefly discuss the steps that need to be taken when such data is required for the operational

needs of maritime software applications. The first step –prior to the data collection– is to identify the classes of data required in order to make the system work efficiently and reliably. Subsequently, a variety of data sources for each kind of data are surveyed towards the selection process. These sources are refined on the basis of a number of predefined criteria to which minimum standards apply in respect of quality and precision, as well as on the basis of the area covered by the data needing to be found. Once usage restrictions (which will be adhered to for the extent of the life circle of the marine information system) for these data have been duly taken into account, the data are collected from their sources and appropriate software tools are adopted or developed towards the efficient integration and storage of these data in the database of the maritime information system.

The backbone storage system can be any of the open access database management systems, from the PostgreSQL and the MySQL to any of the commercial ones, such as the Oracle Database and the Microsoft (MS) SQL Server, necessarily accompanied by their specialized extension subsystems for the efficient handling of the geometric nature of these data. At this point a selection process of the data needs to be initiated in order to eliminate duplicate entries that might occur since data that are collected from heterogeneous sources could partially overlap. There might also be a need to transform the format of the data –*e.g*. from raster to vector or vice-versa– because the manipulation and joining of related datasets that appear to be in different formats is both difficult and time-consuming. The collected data are then safely stored and their optimization for efficient manipulation in the database system is performed: database indices for fast retrieval might need to be built; the responses of users' popular queries for data that are not dynamically changed overtime (*i.e.*, such as coastline static data) might need to be pre-computed and their results stored separately in the database so as to minimize avoidable delays in the future operating of the system, *etc*. Once these steps are taken, the data become available in answer to the requirements of the maritime information system.

3.4. Maritime Geospatial Data Classification

The wide range of nautical or marine data needed to develop and operate an efficient maritime information system, to be analysed further subsequently, falls into one or more of the following wide categories:

- up-to-date geospatial data related to human-life on or near the seas, such as ship traffic data and technical data regarding the several characteristics of ships, data related to maritime areas of particular interest to humans (e.g., harbors, fishing areas), etc.,
- geospatial data related to marine biome and wild-life in/on or around the seas, such as data regarding particularly sensitive maritime areas for wild-life and data for fish and sea animals reproduction areas, etc.,
- annotated data related to accidents history at sea,
- marine meteorological forecasts and climate data,
- nautical cartographic data related to geospatial objects of critical importance in/on or around the seas, etc.

Maritime traffic data are usually transmitted in the form of real-time streams of Automatic Identification System (AIS) messages **[212]**. The benefit of AIS for all mariners lies mainly in its capabilities with regard to increasing navigational awareness, to assisting with avoiding collisions and with the port authorities' more efficient control of maritime traffic.

The technical characteristics of the ships are mostly static or rarely affected by changes. If they are affected, the changes take place under certain conditions (e.g. the type of usage of the ship changes):

- type (passenger, tanker, etc.), size (length, etc.),
- manufacturer, year of manufacture,
- owner/manager, firm, home port, flag,
- fuel consumption, maximum speed, draught,

- type of cargo, weight of cargo, tonnage,
- Froude number (related to wave resistance), several coefficients related to the hull of the ship (e.g., block/midship/prismatic/waterplane coefficients), etc.
- photographs and videos (if they exist) depicting the ship and some of its characteristics, etc.

The geographic regions of marine areas with particular sensitivity and restrictions may include:

- environmentally protected areas (e.g. parks that are strictly protected by laws and legislations),
- significant areas for marine biodiversity (e.g. marine mammals, sea turtles, birds),
- island wetlands and coastal waters surfaces,
- major fishing areas,
- areas where fish farms are located and demarcated areas of organized aquaculture,
- military shooting ranges,
- hazardous shipwrecks (location and depth),
- submersible cables (location and depth), etc.

As regards accidents history at sea, the following data are of great weight:

- exact geographic coordinates and description of marine accidents which have occurred in the past, involving either ships/vessels (e.g., collision or contact or capsizing or grounding of vessels, including the IMO numbers of the ships involved if any), or oil and gas drilling platforms (e.g., explosions, oil spills), or any other on-sea or on-land source around the sea (e.g., marine chemical pollution incidents caused by factories near a coastline), etc.
- up-to-date tracked information for any man-made source near or on the sea with a heavily documented accident history (e.g., ships, oil and gas platforms, underwater pipelines, on-land installations of several types), and with a history of violations or of incidents of non-compliance with international and national maritime regulations,
- incidents involving dangerous ship movements and trajectories on record,
- the list of 'flag of convenience' countries [213] under the protection of which some ships are registered for the purpose of avoiding regulations and tax obligations in the owners' country, etc.

The supply of highly accurate location-based meteorological forecast data in real-time streams is a major factor for efficiency and may include the following parameters:

- weather forecast: surface wind, rain fall, cloud cover, temperature, atmospheric pressure, etc.,
- wave forecast: wave height, swell height, swell period,
- sailing forecast: wind speed and direction, wave height and direction, visibility, etc.,
- sea level forecast: total elevation, tidal elevation,
- sea traffic forecast: surface temperature, salinity, surface movement, free surface elevation,
- ecosystem forecast: chlorophyll, nitrates, phosphates, bacterial biomass, phytoplankton biomass, etc.,
- oil spill, satellite imagery, high-frequency telemetry, etc.

Additional statistical data can also be extracted on the basis of historical weather data in order to analyze the level of influence of every weather data parameter in monitoring the health of the maritime wildlife and climate change (e.g., phytoplankton, temperature, oxygen levels), in safeguarding maritime navigation and transportation, in minimizing the risk of accidents, etc.

Nautical cartographic data related to geospatial objects of critical importance in/on or around the seas may include:

- nautical digital charts containing the land and sea boundaries of countries worldwide; the geographic names for regions near, over and under the sea; etc.,
- bathymetry data in geometry and raster formats,
- ports and harbors (i.e. their exact geographic location and their surface, whether they are in operation, etc.),
- shores, beaches and land along the coastline,
- lighthouses (i.e., their exact geographic location, type, color and lighting periods, size, range, if they are in operation), etc.

Additional advanced knowledge can be indirectly extracted (for example such as in [214] and [215]) from combinations of existing datasets of heterogeneous nature using machine learning [216] and data mining [217] techniques, that for example will identify hazardous regions and ship routes in the sea, suspicious and dangerous vessel movements, traffic patterns and popular highways in the sea, high accident risk areas, areas with strong sea currents, highly polluted areas, etc.

Such data can provide critical information such as the navigation behavior and performance of a ship or of a type of a ship in relation to and depending on the meteorological conditions (inclination, route deviation), the condition of the sea surface and the intensity of sea currents in relation again to the meteorological conditions, the habits of fish and sea animals depending on the season of the year, the impact of human activity on the sea near fish reproduction areas, the sea pollution depending on the weather conditions, and several other related findings depending on their application.

Data class description	Most common data format	Real-time	Current & recent historical	Long term historical
AIS data	AIS RAW messages	√	√	no existing data
Vessels' data	Clear text	–	√	–
Marine biome-related data	Clear text & geospatial	–	√	–
Accidents-related data	Clear text	–	√	√
Weather data	XML	√	√	√
Climate data	Clear text	–	√	√
Nautical cartographic data	Clear text & geospatial	–	√	–

TABLE 23: DATA CLASSES WITH THEIR MOST COMMONLY SEEN FORMATS.

Table 23 summarizes some selected data classes surveyed in this section along with their most commonly seen data formats. The time categorization gives a rough estimation of the lifetime and validity of the importance of the data. The 'Real-time' column refers to data that are continuously updated (stream data), the 'Current & recent historical' column refers to data that could be up to 10 years old, and the 'Long term historical' column refers to data considered to be highly important for most relative applications even many years after their production.

3.5. Data Sources

This section will outline the various resources for collecting maritime related data such as vessel positioning data, weather and climate data, data related to protected and sensitive areas, marine accidents, flags of convenience, port related data, data related to anti-shipping activities, natural hazards, navigational aid systems, cartographic and coastline data, bathymetry data and others.

3.5.1. Vessel Tracking and Monitoring Services

MarineTraffic [218] is the most popular interactive maritime information system developed by the University of the Aegean. Its key objective is the online monitoring of ship movement worldwide,

while providing the public with real-time information about port arrivals and departures. The success of the coverage provided relies on voluntary participation in the community and on local authorities installing receivers and sending the collected data in real-time to the central MarineTraffic server that in turn collects the data and visualizes them on an online map. The data are sent and collected in raw AIS messages format through UDP channels. Additional TCP requests can be performed in order to retrieve data in XML and JSON format. Historical data can be retrieved on demand, using requests in XML format.

Vesseltracker [219] is a provider of AIS vessel movements on the global scale and of maritime information services such as maritime news and events, vessel information, reports and statistics. More specifically, Vesseltracker provides its registered members with customized real-time and historical AIS data; a comprehensive vessels database of specifications, characteristics, equipment, ownership and management information; alerts on vessels status and on customizable regions *via* email, SMS and phone; information and alerts on expected, arrived and departed vessels for a single port or a list of ports; information about the distance to be covered by vessels to ports; port events; map views and layers including nautical charts; global and local piracy information; and weather forecasts information.

MariWeb [220] is a monitoring service for the movement of ships and for other relevant maritime information, such as the characteristics of ships, their destination, estimated time of arrival, photographic data, traffic statistics for ports, *etc*. The platform is developed by the IMIS Global [221] which is a technology-oriented company focusing on offering AIS network management software tools, *i.e.* tools that efficiently collect, store and display AIS data securely providing navigational and fleet monitoring services to its customers in the maritime context. The company uses its own private network of receivers that forward the collected data to the central server of MariWeb for visualization.

ShipFinder [222] and FleetMon [223] are services with characteristics similar to MarineTraffic, Vesseltracker and MariWeb services. Lloyd's List Intelligence [224] is a specialized service of the Lloyd's List Group, dedicated to the global maritime community. Access to the monitoring service of ships is limited to certified members on a subscription basis.

VesselFinder [225] is another popular service that provides visualization of various real-time, time-evolving and static maritime data. The service is developed by the AIS Hub [226] data sharing center which is the only online service worldwide that distributes freely all its real-time collected vessel traffic data to any party volunteering to contribute reliable real-time AIS data to its network, constituting thus a valuable resource for maritime professionals and software applications developers. Its Web service can provide data in XML, JSON or CSV formats *via* TCP requests.

Free-of-charge real-time test data in the form of raw AIS messages can also be retrieved from Exploratorium [227] for non-commercial use. This source is also registered at the MarineTraffic service. It is considered to be an excellent source for educational and software applications development and testing purposes. The real-time raw data come from vessels in the region of the San-Francisco Bay.

Real-time satellite data in the form of raw AIS messages can be retrieved on a subscription basis from the MarineTraffic service [218] or from the IMIS Global [221]. Satellite AIS data guarantee coverage in every sea location on the Earth *via* dedicated hardware the cost of which has meant that vessels are not yet mandatorily equipped with it under the current international legislation. This approach has been developed in order to address the poor quality or absence altogether of AIS coverage in the larger oceans.

3.5.2. Vessels and Shipping Companies Data

Over 160,000 ships, passenger and cargo vessels of 100 tons and over are sailing the seas or are stationed at more than 13,000 ports globally. For this reason, several specialized online databases have been developed, providing accurate, detailed, both historical and current, data related to vessels and shipping companies.

The IMO Numbers Database [228] is a freely accessible -through free registration - database provided by the IMO and which was promoted after 9/11 to enhance the security of vessels and

ports facilities. Every passenger carrier and seagoing vessel of 100 tons or above receives a mandatory and unique IMO number. For every ship identified by its IMO number, the database provides accurate information about its name, its flag, its type, its overall capacity and weight, and its year of manufacture. ShipList [229] provides a free access database of characteristics that partially complement that of the IMO Numbers Database. The service stores important details for every ship but, in practice however, some fields of information often remain void and the data elusive, especially as regards the maximum ship speed, net tonnage, fuel capacity, *etc*. ShipNumber [230] is another freely accessible online source that provides information, for a specific ship name or IMO number, about the ship's flag, call sign, ship type, gross tonnage, dead weight, total length, extreme breadth, draught and year of construction,. VesselFinder [225] is another free access database containing information about the identity, the dimensions and other technical features of the ships. The information partially overlaps with the data that can be retrieved through other similar databases on this list. Equasis [231] is an excellent service that has been developed to become a powerful and reliable tool dedicated to the safety of ships and shipping. The service provides to its registered members free-of-charge details about the history of the ships, the owner companies or consecutive owner companies, inspection, manning, and other categories of data. Very importantly, it also provides information not to be found elsewhere regarding the status of blacklisted ships. Veristar [232] is a database that shares some features with the Equasis database. Overall the service provides free-of-charge information about vessels and shipping companies, about inspection history data and so on. Maritime-Connector [233] is an online database providing historical information about the identity, the manager/owner, the manufacturer and also the safety category/class of every vessel. The IMO [206] also provides, among other information, a number of technical details for every ship travelling on the seas across the globe (it is to be noted that this data source is different from the IMO Numbers Database mentioned earlier). GrossTonage [234] provides its registered members with a free access repository containing technical information about the ships along with a brief description of marine incidents that can be visualized on a map.

3.5.3. Protected and Other Sensitive Areas

The cartographic data of aquatic areas protected by international conventions (BIOGEN, BIOSPHERE, DIPLOMA, MPK, BARCELONA, *etc*.) can be collected from the Protected Planet portal [235], the largest online geographic database of marine protected areas, which has been developed by the agencies that constitute the International Union for Conservation of Nature (IUCN) [236]. IUCN is the oldest and largest global organization for the protection of the environment. The source provides cartographic data in electronic form for various protected areas, national parks, wildlife refuges, island wetlands, *etc*.

A notable example of protected areas is the Natura 2000 network [237], which is a European Ecological Network of designated terrestrial and marine areas hosting natural habitat types and habitats of species that are important at the European level and are thereby protected by European Union (E.U.) laws. The network includes hundreds of special protection areas and sites of communal importance that have already been designated for strict legal protection, with numerous others waiting to be included. The Natura 2000 Network Viewer can be accessed from [238]. The complete and up-to-date Natura 2000 dataset is shared freely by the European Environment Information and Observation Network Central Data Repository [239].

The United Nations Educational, Scientific and Cultural Organization (UNESCO) Geoparks are geographic areas in which sites and landscapes of international geological significance are managed within a holistic concept of protection, education and sustainable development. The list of the geoparks around the globe can be found in [240]. As of at the time of writing, 116 national geoparks from 31 countries and 4 transnational geoparks have been included in the list with many others awaiting inclusion.

Many sources exist which contain data related to biodiversity and wildlife. The Global Biodiversity Information Facility [241] provides free and open access to biodiversity data around the world. The VertNet [242] is a National Science Foundation-funded collaborative project [243] thanks to which biodiversity data is free and available on the Web. The backbone of the VertNet project consists of four individual networks, the MaNIS database with mammals-related data [244], the HerpNet database with amphibians and reptiles-related data [245], the FishNet database with

fish-related data [246], and the ORNIS database with birds-related data [247]. The OBIS [248] is a marine species database repository for the world's oceans, provided by the UNESCO. The users can identify biodiversity hotspots and large-scale ecological patterns, analyze dispersions of species over time and space, and plot species' locations with temperature, salinity, and depth. The World Conservation Monitoring Centre (WCMC) [249] is the specialist biodiversity assessment arm of the United Nations Environment Programme (UNEP) [250]. It provides, among other data about biodiversity such as marine ecoregions and pelagic provinces of the world, global maps of various biodiversity indexes, chlorophyll-a concentration, global distribution of whales, dolphins, seals, turtle nesting and feeding seamounts and knolls, mangroves, *etc*.

The ReefBase [251] is an online collection of all available data and knowledge about coral reefs. The FishBase [252] is the premier biodiversity data website for all the fishes of the world. The Biodiversity Information System for Europe [253] and the European Nature Information System [254] provide data about the species, habitat types and protected sites across Europe, while other data exist that relate to land, water, soil, air, marine, agriculture, forestry, fisheries, tourism, energy, land-use, and transport. The European Marine Observation and Data Network's (EMODnet, [255]) portal for seabed habitats [256] is a free resource for marine habitat data in Europe.

The WWF Conservation Science Data and Tools [257] provides, among other instruments, a toolkit to visualize the global distribution of animal species. The portal also provides a variety of datasets of the Earth's freshwater and terrestrial biodiversity, marine ecoregions, hydrographic data for analysis and planning, biogeographical data of grassland ecosystems, *etc*. The IUCN Red List of Threatened Species [258] contains spatial data and assessments for just over 76,000 species. The portal provides taxonomic, conservation status and distribution information on plants, fungi and animals that have been globally evaluated to determine the relative risk of extinction.

The sensitive area of offshore archaeological sites is acknowledged in the creation of several online resources such as Pleiades [259] which is a gazetteer of ancient on-land and under-water places that provides archeological geospatial data. It offers an extensive coverage of the Greek and Roman world, and is expanding into Ancient Near-Eastern, Byzantine, Celtic, Early Islamic and Early Medieval geography. The Ancient World Mapping Center [260] provides free maps with data related elevation tints, labels, point symbols and shaded relief for the Roman Empire, the Byzantium, the Aegaeum Mare, the Iberian Peninsula, as well as about aqueducts, inland waters and cultural geography metadata, such as Greek geographic names in Greek nominative forms (in UTF-8). The Greek archeological cadastre [261] and the American School of Classical Studies at Athens [262] provide archeological geospatial data on the ancient Greek world, such as a large number of ancient cities and locations, rivers, elevation data, and so on. A digital Atlas of the Roman Empire can be found at [263]. The Pelagios Commons [264] is a community and online resource for linked open geodata in the Humanities. Its Peripleo service is a map-based search engine for exploring archaeological, textual and image-based data that has been annotated by the Pelagios community. Its Recogito service is a Web-based tool that makes it easy to identify and record the places referred to in historical texts, maps and tables.

World Heritage sites (cities, islands, lakes, mountains, *etc*.) are listed by the UNESCO and an extensive list can be found in [265]. The dataset can be downloaded as an MS Excel or XML or KML document. At the time of writing, 1031 sites from 163 countries that have signed and ratified the World Heritage Conventions are included in the list, most of which appear on it on the basis of culture and nature criteria. The Managing Cultural Heritage Underwater project [266] provides a tool to exchange and explore underwater cultural heritage information.

The Ocean Energy Systems initiative [267] offers an interactive map of global offshore marine energy facilities and resources. The WindFarm Action Group [268] provides the location and other related information of onshore and offshore wind farms.

The Greg's Cable Map [269] is an interactive map with data related to currently active or planned undersea telecommunication cables. The dataset can be retrieved in raw, KML or ArcGIS format. Similarly, the TeleGeography Submarine Cable Map [270] provides information about submarine telecommunication cables and their landing points. On the Subsea Cables Consultants Ltd [271] site a number of maps can be found that represent the location of several submarine power cables on the global level. The International Cable Protection Committee portal [272]

https://www.iscpc.org/cable-data/ provides an up-to-date database of information relating to the majority of active and planned international submarine telecommunication, power and scientific cables, *i.e.*, cables for scientific research purposes (*e.g*. oceanographic or seismic). The Kis-Orca interactive map [273] is an offshore renewables and cables awareness project that in addition provides data about offshore power cables, oils and gas pipes and renewable energy construction on the UK territory.

Because of the impact on the maritime environment of any malfunction, the EMODnet's human activities marine portal provides oil rig and gas rig data, boreholes and offshore drilling sites with their locations [274]. The Peace Research Institute Oslo (PRIO) network [275] hosts a petroleum dataset containing data concerning all known onshore and offshore oil and gas deposits along with additional potentially relevant data about diamond resources, length of international boundaries, shared rivers and other non-geographic data such as economic and socio-demographic and warfare data. The U.S. Geological Survey (USGS) World Petroleum Assessment [276] provides information pertaining to the 2012 assessment of undiscovered, technically recoverable conventional world oil and gas resources. The Theodora World Pipelines maps website [277] provides information about the diameter, length and capacity of several crude oil (petroleum) and natural gas pipeline installations across the globe.

3.5.4. Marine Accidents

On the national level, Maritime accidents history data can be retrieved from government agencies, such as the Search & Rescue Department or the Maritime Security Department, if they exist or, alternatively, from other agencies under the supervision of relevant national ministries and governmental departments dealing with the merchant navy and the maritime domain or the environmental protection domain, *etc*. Whenever data history is provided on the condition that it is not to be published, its value lies in the possibility of extracting useful knowledge about hazardous areas and vessel routes and trajectories, and about ships and man-made above-the-sea-surface installations with a documented accident history. Some of these datasets may only provide approximate descriptions of the site of the incident, rather than the exact location. While some of these descriptions define specific bounded areas (*e.g*. '2NM West of Heraklion, Crete') others might refer to less defined areas (*e.g*. 'in the sea area on the east of the Mauritius Island'), therefore a user would need to geocode this information in order to extract the relative geographic coordinates or the estimated wider region in which the accident took place.

In several countries, organizations such as the National Bureau for Marine Casualties Investigation Organization (for example, in [278] and in [279]) might be able to provide information about a number of marine accidents that have taken place on the country's territory and also to provide detailed investigation reports for every accident. Statistics derived from the accidents can be provided along with the national and international legislation related to marine accidents.

Lists recording vessel accidents can also be freely available on the Web. An example is the list of 114 marine accidents available from the U.S. National Transportation Safety Board [280]. For every accident the list includes its geographic position and an analysis of the various parameters that caused it. Another list of seven accidents accompanied by analyses of the ships' routes prior to accidents is available at [281]. Also, a detailed list of major maritime accidents that have been recorded across the world since the year 1120 AD can be found on [282]. The analysis (*via* links to Wikipedia pages with details for every accident) also includes the geographical location in which every one of these accidents occurred. Another detailed list of accidents involving ships on the international scale is available from the IMO [206]. The accidents data provided by the IMO can be obtained free-of-charge on the condition that they will be strictly used for non-commercial purposes. The WreckSite [283] database provides extensive online information about 163,020 shipwrecks worldwide, including data such as geographical location, ship details, images, owners and builders, maritime charts, *etc*. A list to access the national accident investigation reports from 24 counties is provided on the Marine Accident Investigator's International Forum [284]. Also, the Maritime Bulletin [285] provides a list of marine accidents around the globe, along with piracy reports and weekly reports regarding shipping hazards. The European Maritime Safety Agency in [286] provides summaries and safety recommendations from marine investigation reports, as they have been compiled by the competent authorities of E.U. Member States.

Data relating specifically to oil spill accidents can be retrieved freely online, for example from the International Tanker Owners Pollution Federation (ITOPF) portal [287]. The source also provides additional information for the 20 most catastrophic oil spills since 1970. The IncidentNews website [288] of the U.S. National Oceanic and Atmospheric Administration (NOAA) [289] provides abundant data about selected oil spills (and other incidents) for which the NOAA's Office of Response and Restoration (OR&R) [290] provided scientific response-support for the incident. The software and datasets publications, training, and othe resources of the NOAA's OR&R are dedicated to environmental restoration and provide response tools for oil and chemicals spills and marine debris.

3.5.5. Flags of Convenience

A renewable list [291] of 26 countries with a flag of convenience [213] has been compiled by the Fair Practices Committee of the International Transport Workers Federation (*i.e.* a joint committee of the federation of seafarers and dockers unions) and a slightly different list including a few more countries can be found in [292].

3.5.6. Port State Control Data

Port state control [293] refers to the inspection of foreign ships in national ports to verify their compliance with the requirements of international regulations and rules. Nine regional agreements exist on state control of ports, or Memorandum of Understanding (MoU). A list of nine online databases that contain all vessels currently detained by regional authorities around the globe can be retrieved from the following sources: for Europe and the north Atlantic (Paris MoU) [294]; for Asia and Pacific (Tokyo MoU) [295]; for Latin America (Acuerdo de Viña del Mar) [296]; for the Caribbean (Caribbean MoU) [297]; for West and Central Africa (Abuja MoU) [298]; for the Black Sea (Black Sea MoU) [299]; for the Mediterranean (Mediterranean MoU) [300]; for the Indian Ocean (Indian Ocean MoU) [301]; and for the Riyadh MoU [302]. U.S. port state control is carried out by the U.S. Coast Guard [303].

Data about the ships complying with the regional regulations of the Paris MoU, Tokyo MoU and the U.S. Coast Guard Port State Control can be retrieved from the Equasis portal [231].

3.5.7. Anti-shipping Activities

The U.S. National Geospatial Intelligence Agency provides freely an up-to-date spatial dataset of more than 7,000 anti-shipping activity messages [304]. The dataset includes the exact geographical location and description of specific hostile acts against ships and mariners from 1985 until today. These data can be useful for the recognition, prevention and avoidance of potential hostile activity in the future. The dataset can be downloaded as a KMZ file, ESRI shapefile or as a personal Geodatabase in MS Access database format.

3.5.8. Nautical Weather Forecast and Climate Data

Several services provide meteorological data *via* an application programming interface (API) which allows researchers and developers to access weather forecast conditions for both land and sea. The services provide data for temperature, any precipitation or presence of fog, speed and wind direction, the height of sea waves, the direction of sea waves, and include weather description icons, *etc*.

Some services make the data available for free for personal use or for empirical purposes, while others allow the development of applications for commercial use. The data are usually available in XML or JSON. An extensive list of 76 relevant services *via* API can be found in [305].

Nautical or marine meteorological forecast data for research and development purposes tends to be scarce. In shipping applications, detailed weather forecast data of up to six to seven days ahead is important, hence a brief reference to some of the few providers of free-of-charge meteorological forecast data for marine applications.

World Weather Online [306] provides land and marine meteorological forecast data through the use of a free account and a specific API key. The marine forecast data have a time window of 24

hours, regularly updated and covering a time span of 6-8 hours and include: temperature, humidity, visibility, cloud cover, wind speed, wave height, swell height, precipitation, pressure, *etc*. The user of the free service can obtain meteorological forecasts for up to 500 requests per hour and no more frequently than every 15 minutes for the same location. Land weather forecast data are updated every 3-4 hours. The available information is sent to the user *via* XML, JSON and CSV format. The data request has to be accompanied by the longitude and latitude of the relevant location on the Earth. The meteorological forecast is retrieved from the nearest weather station to that location. World Weather Online also provides weather and tidal data history.

Weather Underground [307] provides detailed meteorological forecast data per hour for the following 24 hours and as well as a prediction of the weather for the next three to ten days ahead, together with dynamic animated satellite images. The available data are accessed *via* API and can be obtained in the JSON or XML form. Rather more restricted than the Word Weather Online service, Weather Underground provides free usage of the service for up to 10 times per minute and up to 500 times per day.

The Severe Weather Information Center [308] provides global warnings about tropical cyclones, heavy rain, snow, thunderstorms, gales and fog. Its equivalent system for Europe is named Meteoalarm [309]. The Arizona State University [310] holds an interactive map and an archive for extreme global weather and climate conditions.

The NASA Earth Observatory [311] provides 16 global animated maps and datasets which are related to weather and climate conditions such as sea and land surface temperature and anomalies, rainfall, snow cover, *etc*. over a time span of 12 months.

Daily reports acquired through sensors about air pollution and ozone can be retrieved on the national level from environmental agencies or from the ministry of the environment of a country concerned, such as in [312] which provides live and historical data. Additional information about air quality, air pollutants and emissions can be retrieved from such sources, as for example the Air Quality database [313] which is provided by the European Environment Agency (EEA) [314].

And last, there are online databases that offer historical weather and climate data and some useful statistics:

- the collection of global daily measurements of weather features (temperature, wind speed, humidity, pressure, etc.) for the period 1929-2009, from over 9,000 meteorological stations around the world, which data have been uploaded from Infochimps.org onto the Amazon Web Services [315];
- the Climatological Database for the World's Oceans [316] which is based on the climatic data contained in ship logbooks for the period 1750 to 1850;
- the European Climate Assessment and Dataset project collection [317] which provides historical data for the period January 1, 1950 - December 31, 2012; etc.

3.5.9. Natural Hazards

Because of their close relation to tsunami phenomena, seismic and volcanic activity monitoring is crucial. Such data are usually provided in almost-real-time by the institutes for geodynamics on the national and international levels, such as [318] and correspondingly [319], the latter a globally-recognized creditable online center for almost-real-time information for European-Mediterranean earthquakes and for worldwide earthquakes with M4.0+. Earthquakes and waveform data for Europe are also provided by the GEOFON Program [320] and for the U.S. by the USGS Earthquake Hazards Program [321]. Both sites provide almost-real-time data feeds.

The International Tsunami Information Center [322] provides international tsunami warnings and contains further data related to seismic activity and sea level stations along with historical data. The Global Risk Data Platform on Natural Events [323] covers data related to tropical cyclones, storms, surges, drought, earthquakes, biomass fires, floods, landslides, tsunamis and volcanic eruptions. And last, Volcanoes of the World [324] is an online database describing the physical characteristics of volcanoes and their eruptions.

3.5.10. Navigational Aid Systems

A database of international navigational aid systems around the globe can be retrieved *via* the MarineTraffic service [218], with every entry containing information related to the name of the navigational aid system, its location, a representative photo, its type, its range, color of light, flash duration, time interval of operation, and whether or not it is active. The geospatial sea surface region covered by a navigational aid system can be computed by its range, taking into account the local coastline.

3.5.11. Sea Ports Locations and Facilities

The U.S. National Geospatial Intelligence Agency provides the geographical locations and characteristics of the ports around the Earth through the dataset "The World Port Index (Pub150)" [325], which keeps a record of the locations of 3,717 ports worldwide. The dataset is provided free-of-charge for non-commercial use in an MS Access database or in an ESRI shapefile, and provides useful detailed technical information such as the size, type, anchor depth and tidal range of harbors, their fuel/oil supply facilities, available repairs support, and much more. Wikimapia [326] provides similar data in KML geoformat that includes 10,478 port facilities worldwide. No other important technical information about the available ports is provided by this particular dataset.

Additional information about ports worldwide can be found in VesselFinder [225]. This database contains the name of the port, the country in which it is located, its size (in such categories as: small, medium-sized port, *etc*.) and its geographic position on the map. Information about ports can also be found in MarineTraffic [218], including the name and location of every port, together with real-time data about the presence of vessels and the expected arrival and departure times into and out of the ports.

Finally, data about airports, runways, airlines, radio navigation aids and waypoints that can become relevant in the domain of a maritime GIS application can be found at OurAirports [327], OpenFlights [328], and WELT2000 [329].

3.5.12. Essential Naval Cartographic Data

Borders between countries, while they are mainly static data, are occasionally updated when affected by changes. Such a dataset can be retrieved from several online sources in various formats and sizes (*i.e.* resolution). An excellent source for this dataset is the Blue Marble Geographics [330] which provides country boundaries in ESRI shapefile or TBA files format.

An unclassified vector-based digital dataset compiled from a portfolio of approximately 5,000 nautical charts and containing the boundaries of countries worldwide enriched with several additional maritime features is provided free-of-charge by the Digital Nautical Chart portal of the U.S. National Geospatial-Intelligence Agency [331]. It is available in 29 subsets of data divided by the region of the planet to which they correspond.

3.5.13. Maritime Borders, Coastline and Land Areas

The coastline (or shoreline) is an important dataset that defines which areas of the Earth are land and which are ocean or sea. Several online sources provide global coastline datasets in different resolution and formats. An excellent example is the high resolution and complete (without gaps due to missing data) related dataset which is provided by the U.S. NOAA Shoreline website [332]. A dataset of coastlines of the world in ESRI shapefile format which the U.S. Defense Mapping Agency developed from various sources is also provided by the Pacific Disaster Center in Hawaii [333]. Finally, the global coastline with the exception of Antarctica, can be retrieved in several resolutions (full, high, intermediate, low and crude) and formats (ESRI shapefile and native binary files) from the GSHHG database [334], which is a global geographic database that is kept constantly updated.

Many organizations and projects produce and distribute high resolution national or continental coastline data. Focusing for example on the European continent, the geographic data of the

European coastline [335] and the European maritime borders [336] that have been produced in the context of the EUROSION project [337] derived from the United Nations Convention for the Law on Sea, can be retrieved from the list of datasets in [338] which are provided by the EEA [314], which is responsible for the independent provision of information relating to the environment. High-resolution data for the European Coastline (and land surfaces) [335] may also be obtained from the EU-Hydro, which is a set of hydrological data developed under the program Copernicus [201], the largest scale Earth observation program. Interested parties might find it worthwhile to give priority to the first of the above-mentioned geospatial datasets (*i.e.*, that of the EEA), and to consider the second (*i.e.*, that of the EU-Hydro) as a support dataset because the coastline from the EU-Hydro represents the separation between land and sea, as indicated by satellite images of the dataset in [339], provided by the European Space Agency. The tidal data depend on the date and time when the images were taken, hence the dataset's insufficiency in respect of the requirements to define the coastline.

3.5.14. Naval Bathymetry Data Maps

Bathymetric data can be retrieved from various heterogeneous sources. The datasets may share overlapping information in different data formats or precision, therefore appropriate data transformation and refining processes may need to be undertaken prior to the process of integrating the data into the same database. Important data sources that can be accessed online are:

The Marine Geology & Geophysics and the Bathymetry & Global Relief discipline of the National Center for Environmental Information [340] of the NOAA [289] provides access to sonar data (single-beam trackline bathymetry surveys), magnetic, seismic and other data [341] that have been collected on the basis of marine survey trips since 1939 until today. The source also provides rich multi-beam sonar bathymetry data [342] that contain over 1,187 international marine trips that collected bathymetry data from several areas around the world.

The International Hydrographic Organization (IHO) [343] collects and quality-checks globally oceanic sounding data acquired by hydrographic, oceanographic and other vessels during surveys or while on passage. The IHO members have also made additional contributions with shallow water sonar data derived from electronic nautical charts.

The International General Bathymetric Chart of the Oceans Cooperation [344], [345] which is maintained by the British Oceanographic Data Center (BODC) [346] provides free-of-charge bathymetry data that are related to all the seas across the globe. The data are collected by echo-sounding and the dataset is enhanced with satellite data.

Another notable example of high-precision bathymetry data is offered freely by the EMODnet bathymetry portal [347] of the European Marine Observation and Data Network which contributes to the provision of reliable and interoperable marine data in public and private organizations. The bathymetry dataset of the EMODnet service covers a wide range of marine areas across and around Europe, providing highly accurate Digital Imaging Modeling Soil data and substantial coverage for the corresponding seas.

3.5.15. Tides, Eddies and Sea Levels

The NOAA Center for Operational Oceanographic Products and Services [348] has gathered oceanographic data for over 200 years, serving both the public and government agencies. Its data include sea-level measurements over time; tide prediction locations; a history of currents activity at different levels of depth; *etc*. The dataset can be downloaded in KML format from [349].

Sea-level data on a worldwide scale can also be retrieved from the Sea-Level Station Monitoring Facility [350] which utilizes an interactive map illustrating the locations of stations that measure the sea-level in real-time. A disclaimer on the portal indicates that quality control has not been applied to the data on display and that they are provided as received.

The mesoscale ocean eddies are currents that transport heat, salt, energy, and nutrients across the world seas. Their accurate identification and tracking is crucial for understanding future marine

and terrestrial ecosystems and their sustainability. A rich historical dataset of oceans eddies from 1992 to 2011 can be found in [205].

3.5.16. Various Other Geospatial Data

The Global Earth Observation System of Systems (GEOSS) portal [351] provides worldwide data in relation to water, ecosystems, agriculture, climate, natural disasters, *etc*. A number of datasets in raster and vector format related to land-cover; administrative boundaries; parks; hydrology and ocean water; drainages with lakes; *etc*. on several scales of resolution (large, medium and small) is provided by Natural Earth [352]. The NASA's Earth Observation System provides Earth science data that can be retrieved from [353]. A large number of GIS datasets regarding elevation, transportation, demographics, environment, imagery, water, *etc*., for almost all the countries around the globe is provided by the Massachusetts Institute of Technology (MIT) Geodata Repository [354].

The NOAA's Office of Coast Survey [355] offers a large number of links for U.S. national charts, surveys wrecks, historical data and other useful nautical GIS information. In [356] several datasets relating to administrative boundaries, biological data, climate, land-cover, *etc*. are provided for the U.S. region. The EEA hosts more than a hundred sets of environmental data related to the E.U. territory in [357]. The same portal also hosts a large number of related maps and informational graphs. Another large number of related datasets is hosted by the European Space Agency in [358]. In [359], a number of datasets relating to land-cover, elevation, hydrography, protected sites and other data for the E.U. territory is provided by the EuroStat European Statistics Organization. The BODC [346] distributes biological, chemical, physical and geophysical marine data and also hosts various related portals and project websites. Most of the data maintained and re-distributed by the BODC are not restricted to the U.K. territory.

The Marine Plan website [360] provides a number of datasets that include the major regional fisheries areas controlled by governing bodies across the globe; the state of fish stocks; waters under the sovereignty of countries in other regions of the world (this includes the waters of outermost regions and of overseas territories); territorial disputes conflict zones, piracy hazards; international straits and channels; oceans and continents; nuclear marine areas, oil and gas in the world; *etc*.

A large number of links to worldwide marine and coastal GIS data and image portals is provided in [361]. An extensive framework for sharing world maps and digital geospatial data about the Earth's frozen regions (including snow cover, sea ice extent and concentration, glaciers, ice sheets, permafrost, and other critical components of the Earth's cryosphere) can be retrieved from the Atlas of the Cryosphere [362] in image, GML and GeoTiff formats. Also, the Quantarctica raster datasets [363] include geographical, glaciological and geophysical data for the region of Antarctica.

DIVA-GIS [364] is primarily a free and open-access software tool for data mapping and geographic analysis. Its corresponding portal also provides freely available spatial data, either at the country level (such as administrative boundaries, inland water, roads, railroads, altitude, land-cover, population density, *etc*.) for any country in the world, or at the worldwide level (such as high resolution satellite images, global boundaries between countries, global climate data, species occurrence data, *etc*.). The data can be used in DIVA-GIS and other software tools.

A list of over twenty useful links to services databases for the marine science community, with resources such as abstracts, bibliographies, glossaries and directories, as well as conference proceedings papers which are otherwise not available online are hosted by the Hellenic Centre for Marine Research [365]. Among them and of particular interest is the link to the Institute of Oceanography of the Hellenic Centre for Marine Research [366] which provides access to its Online Search and Download Service database [367] in relation to physical, chemical and biological parameters in the European and international waters; the European Directory of Marine Environmental Datasets [368] which contains datasets collected by Hellenic scientific laboratories, research institutes, universities, *etc*.; and the EDIOS database [369] which provides measuring and monitoring data with regard to sea observation in the Eastern Mediterranean and the Black Sea.

Another categorized list of links to over three hundred portals providing freely available geographic datasets can be found on FreeGISData [370]. The datasets are related to physical geography (weather and climate, rivers, lakes, elevation, hydrology, *etc*.) and human geography (land-use, wars, population, *etc*.) worldwide, while individual datasets for specific areas or countries are also available.

FreeGIS [371] is a blogspot which also offers links and detailed descriptions of numerous portals which provide free and open-access GIS software; remote sensing; and spatial and hydrology data. The following sources [372], [373], [374], and [375] direct the reader to some data sources worth noting, which provide maps, articles of international interest, *etc*., that are also relevant to this study.

Finally, the GEOnet Names Server (GNS) [376] is a repository of standard spellings of global geographic names for regions near, over and under the seas.

3.5.17. Satellite Imagery

Copernicus [201] is a European GIS for monitoring the Earth. It consists of a complex set of systems which collect data from multiple sources, such as Earth observation satellites, ground stations and airborne and sea-borne sensors. Copernicus services address six main thematic areas: land, marine, atmosphere, climate change, emergency management and security [377]. The collected data can be accessed by performing a free registration to the Copernicus Sentinels Scientific Data Hub [378] which at the time of writing contains Sentinel-1 and Sentinel-2 satellites data. Direct access to the data, along with additional information about the various Sentinel satellites missions can be retrieved *via* the European Space Agency (ESA)'s Sentinel Online portal [379] or *via* its data mirror site at [380]. Satellite data for a number of ESA's missions dedicated to Earth observation can be found in [381].

And last, the eoPortal Directory [382] offers a database of an extensive list of past, operational and future spaceborne missions and a complementary database of several flight missions and projects involving airborne sensors.

3.5.18. Sources that Reach Beyond the Maritime Domain

The previous sections report on sources and repositories that provide maritime and maritime-related data. However, a large number of sources exists with a wider variety of data that reach well beyond the maritime domain but which can, in some circumstances, become relevant in a maritime GIS application. Among these, one of the most popular sources is the collaborative mapping project OpenStreetMap [383], which also offers free-of-charge data about roads; trails; cafés; railway stations; *etc*. In a similar way, Wikimapia [326] provides data about roads; railroads; rivers and ferry lines; various types of points and areas of interest such as parks, villages, cities; *etc*. The GeoCommons Archive [384] is a community-contributed collection of hundreds of thousands of open datasets from around the world. OpenEI [385] provides energy datasets on hundreds of topics, crowdsourced from industry and government agencies in relation to energy efficiency, consumption, demand, and much more.

International and national public open data portals are also great sources for continental and governmental or country-specific data. For example, the INSPIRE Geo-portal [386] provides the means to search and access open geographic data provided by European governmental, commercial, and non-commercial organizations within the framework of the E.U. INSPIRE Directive [387], which aims at the creation of an E.U. spatial data infrastructure. The EEA's data and maps repository [388] of the E.U. provides sound and independent data on the environment. Some major datasets on this repository are related to air, water, land, biodiversity, climate change, noise, *etc.* [389]. A Web map service for the repository is available in [390], while code and APIs (divided into specific topics) for developing GIS applications are available in [391].

The European Data Portal [392] harvests the metadata of public sector information available on public data portals across E.U. countries and offers governmental open data that were collected, produced or paid for by the public bodies and public sectors. The E.U. Open Data Portal [393] is the single point of access to a growing range of data from the institutional and other bodies of the E.U.. The portal is not restricted to GIS data and thus some of its main subjects are employment and working conditions, economics, finance, trade, industry, education, science, *etc*.

On a national level, a governmental source paradigm for providing geographic-related data that also serves as a compliant data infrastructure center at a continental level (for example, for the E.U. INSPIRE and for the European Data portals), is the geospatial open repository of the territory of Greece [394], which offers open data for topics related to biodiversity, water, the environment, the economy, elevation, transportation, health, planning cadastre, social dimensions and many more. The data provided by this source can be complemented by the central library of public data [395] which offers access to the databases of the country's governmental bodies. Several other countries (such as Ireland in [396], *etc*.) offer similar data repositories, and it is expected that, in the years to come, every country (and possibly every municipal sector and region in the country) will offer similar services to the public.

The EMODnet [255] network is a Joint European Coastal Mapping Programme which consists of more than 100 organizations assembling quality-controlled and expert-validated marine data that are offered through the EMODnet portal, such as data on bathymetry (water depth), coastlines, and geographical locations of underwater features and wrecks; data on seabed substrate, sea-floor geology, coastal behavior, geological events and minerals; data on modeled seabed habitats based on seadbed substrate, energy, biological zone, and salinity; data on the concentration of nutrients, organic matter, pesticides, heavy metals, radionuclides and antifouling in water, sediment and biota; data on the temporal and spatial distribution of species abundance and biomass from several taxa; data on salinity, temperature, waves, currents, sea-level, light attenuation, and FerryBoxes (*i.e*., kits with instruments that are placed on board to commercial ships such as ferries in order to monitor the temperature, salinity and other water properties); data on the intensity and spatial extent of human activities at sea; *etc*.

The EUMETSAT Product Navigator [397] is the catalogue of satellite data and products with regard to weather, climate and the environment that are offered in near real-time by the European Organization for the Exploitation of Meteorological Satellites (EUMETSAT) [398].

GEOSS links Earth observation resources worldwide across multiple societal benefit areas. The GEOSS portal [351] provides data about biodiversity and ecosystem sustainability; disaster resilience; energy and mineral resources management; food security and sustainable agriculture; infrastructure and transportation management; public health surveillance; sustainable urban development; water resources management; *etc*.

The Red List of Ecosystems (RLE) [399] of the International Union for Conservation of Nature [400] evaluates the conservation status of ecosystems around the globe. The RLE provides the ecosystem locations information and valuable assessments on the basis of a protocol which includes criteria for assessing the risk of an ecosystem collapse and several categories of risk for every ecosystem.

NASA is a great resource for geospatial data, not only in respect of space exploration but also in respect of the Earth. The numerous domains it encompasses provide the material for the creation of an enormous data center. The EarthData portal [401] provides a variety of Earth-related data. A worldview map of those data can be accessed *via* [402]. The NASA's Earth Observing System [403] is a coordinated series of polar-orbiting and low inclination satellites for long-term global observations of the land surface, biosphere, solid earth, atmosphere, and oceans which consists of the NASA's Earth Observations database [404], the NASA's Earth Observatory database [405], the NASA Visible Earth database [406] and other repositories, every one of which provide a variety of data. The Earth Science Projects Division [407] manages the missions which advance the understanding of Earth. The NASA's Science portal [408] provides data of NASA missions related to the Earth as well as several links to other related portals.

The NOAA [289] as well as the USGS Earth Explorer [409] portals are both great resources for various datasets in numerous disciplines, especially but not exclusively in relation to the U.S. territory.

Natural Earth [352] provides also geospatial data related to populated places; disputed areas and breakaway regions; glaciated areas and Antarctic ice shelves; cross-blended hypsometric tints; grayscale shaded relief of land areas; worldwide terrain depicted monochromatically in shades of gray; *etc*. The ESRI Data & Maps portal [410] contains data across the globe, such as country boundaries, aquatic areas roads, railroads, major cities, topography, bathymetry, population,

gross domestic product, night time views of the Earth, *etc*. The PRIO network [275] contains also some unique spatial and non-spatial datasets of specific interest, such as diamond resources, shared rivers between neighboring countries and data related to armed conflicts [411].

Data source	Land & marine data	Atmosphere & climate data	Governmental & human activities data	Cultural data
Copernicus [201]	√	√	–	–
EMODnet [255]	√	√	√	–
NOAA [289]	√	√	–	–
FreeGISData [370]	√	√	√	√
Natural Earth [159]	√	–	–	√
EEA [314]	√	√	√	√
European Data Portal [392]	√	√	√	√
E.U. Open Data Portal [393]	√	√	√	√

TABLE 24: MOST NOTABLE DATA SOURCES THAT REACH BEYOND THE MARTIME DOMAIN.

Table 24 outlines the most notable sources and repositories containing rich data that reach beyond the maritime domain and are mentioned in this study, along with the most characteristic examples of types of data that they provide.

3.5.19. Marine Conservation Organizations

Hundreds of non-profit and non-governmental marine institutes and organizations (such as the ones listed in [412] and [413]) work either independently or by forming societies and coalitions (such as the Deep Sea Conservation Coalition [414]) on marine conservation and other environmental issues such as biodiversity and global warming. These organizations are committed to researching and to ensuring the protection of the marine environment, and pursuing terrestrial wildlife conservation and they are actively involved in lowering the risk of accidents on and near the sea, and in providing statistics, scientific research reports, and case studies data analysis.

Much of this extensive and high-quality work is made available to the public and can therefore be utilized to enrich maritime information systems. For example, in [415] several datasets are provided for areas that are environmentally and economically sensitive to oil and other hazardous materials spills, areas to which sailing restrictions apply, areas to be avoided, shallow banks, rivers, lakes, manatee population locations, sea turtles nests locations, submerged shipwrecks and other obstructions in coastal waters, public access boat ramps, color aerial photographs, coast guard facilities *etc*., throughout the State of Florida and, more widely, the U.S.. The data are published by the marine conservation Florida Fish and Wildlife Research Institute which works for the protection of the sea across the entire South East of the U.S..

3.5.20. Restrictions Applying to Use of Data

This section discusses the various types of restrictions applying to the use of data, established by the sources providing these data in order to protect the rights of the owners over the data that are made available to inspect and download for the purpose of maritime applications.

Datasets acquired by ministries and governmental agencies and other organizations can be accessed free-of-charge by the public for any use at any time, irrespective of commercial use purposes; alternatively, the datasets might be strictly confidential, which means that while they may remain available for data mining, they may not be published under any condition. A number of sources might allow a degree of use of their datasets in combination with a license of Creative Commons [416] attribution, which means that the grantee of the data will need to indicate their source in the applications in which they are used and will need to take into account some other restrictions as well, those that define the extent to which the data can be copied, distributed, edited, remixed, and built upon, all within the boundaries of copyright law.

The detailed data for vessels and shipping companies from the IMO Numbers Database [228], the ShipList [229], the Maritime-Connector [233], and the VesselFinder [225] services can be

collected *via* their freely accessible online databases. The data, however, cannot be re-published although they remain useful resources to ensure the efficiency of maritime information systems. The data from the Equasis service [231] cannot circulate freely on the Web and can only be accessed freely by registered members.

The vessels accident data that can be obtained *via* the IMO service [206] requires, in return for free-of-charge access, the creation of an account and can be used within the limits of the restrictions imposed on the service.

The use of the datasets of maritime protected areas that can be found online on the Protected Planet portal [235] is limited to use that is of a non-commercial nature [417] and cannot be obtained without prior written authorisation of the UNEP-WCMC [249] which is the data supplier. The same exclusively non-commercial use restrictions of [418] hold also for several other online data banks provided by the UNEP-WCMC, such as for the Ocean Data Viewer [419], *etc*.

In the terms of use of the data [420], the World Weather Online service [306] makes it clear that researchers and developers must not share their API key with other users. The terms of use also indicate that the data are protected by strict copyright and must not be distributed, modified or reproduced in part or in whole, without the prior written authorization of the service.

Most of the types of bathymetry data that can be collected from the data sources as mentioned above cannot, according to the restrictions applying, be used for navigational purposes and, when they can be used, it is for personal use only (a free account is required for this). However, the largest and most precise bathymetry dataset that can be found online which, as has already been mentioned, can be retrieved from the EMODnet bathymetry portal [347], is not accompanied by restrictions of use. While it is indicated that this voluminous dataset is available to the public; for legal reasons, however, its source indicates that the data may not be used for navigation purposes.

The port state control data related with most of the MoUs, and specifically for the Paris MoU [294], Tokyo MoU [295], Black Sea MoU [299], Caribbean MoU [297], Abuja MoU [298], and Acuerdo de Viña del Mar [296] state that the data should not be used for any commercial purpose, reproduced in any other sites, stored in a retrieval system, or transmitted in any form, or by any means, without the prior authorization in writing from the owners of the data.

The use of the datasets of global, continental and national coastlines in most of the sources is provided free-of-charge on condition that the source of the data is mentioned. One example is the dataset of the European coastline [421] that is provided by the EEA [314] which is its copyright owner. Also, the maritime borders provided online within the framework of the EUROSION project [337] are available free of restrictions.

License	Linking	Distribution	Modification
CC-0	Public domain	Public domain	Public domain
CC-BY	Permissive	Permissive	Permissive
CC-BY-SA	Copyleft	Copyleft	Copyleft
CC-BY-NC	Non-commercial	Non-commercial	Non-commercial
GPLv3	With restrictions	Copyleft	Copyleft
ODbL	Copyleft	Copyleft	Copyleft

TABLE 25: THE MOST-COMMONLY-USED LICENSES FOR FREE AND OPEN-SOURCE DATA[2].

Table 25 outlines the most-commonly-used published licenses for free and open-source data, and their restrictions on linking, distributing and modifying the data [422].

The data on the Greg's Cable Map [269] are provided under the GNU General Public License v3 (GPLv3). The USGS World Petroleum Assessment [276] requires copyright permissions [423]. The owner of the archeological data provided in [262] states that the data are offered under the Creative Commons CC-0 licensing. Pleiades [259] states that sharing and remixing data is

[2] A 'Public domain' label states that there is absolutely no ownership such as copyright, trademark, or patent. A 'Permissive' license has some limited requirements, such as crediting the original authors. A 'Copyleft' license permits people to freely copy, modify and redistribute the data as long as they do not keep others from also having the same rights.

permitted under the terms of the Creative Commons Attribution 3.0 (CC-BY) License. The ReefBase [251] states that the data may be used for non-commercial purposes, including research, education, presentations, and non-commercial publication [424]. The FishBase [252] states that this work is licensed under a Creative Commons Attribution-NonCommercial 3.0 Unported (CC-BY-NC) License. The OBIS [248] makes the data available under the Creative Commons licenses CC-0 or CC-BY or CC-BY-NC [425]. The IUCN Red List of Threatened Species [258] states that the data is made freely available to the public for non-commercial use. The Global Risk Data Platform on Natural Events [323] states that all rights are reserved and none of the materials provided on the website may be used, reproduced or transmitted without permission in writing from the publisher [426]. The Quantarctica datasets [363] are free for non-commercial use. The data from the Copernicus Sentinels Scientific Data Hub [378] are offered to the public for free, except when the E.U. law allows for specific limitations of access and use in the rare cases of security concerns, protection of third party rights or risk of service disruption. The ESA's Sentinel full online terms and conditions can be retrieved from [427]. The OpenStreetMap open data [383] provided by the OpenStreetMap Foundation [428] are licensed under the Open Data Commons Open Database License (ODbL 1.0). The WELT2000 [329] database is also made available under the ODbL 1.0 license. The data from Wikimapia [326] are provided under the Creative Commons License Attribution-ShareAlike (CC BY-SA). The E.U. Open Data [393] portal provides the data for free for use and reproduction for commercial or non-commercial purposes. The Theodora World Pipelines maps [277] states that all the rights on the data are reserved by its sponsored Information Technology Associates Company. The ESRI Data & Maps [410] data usage policy can be viewed in [429]. For the data hosted by the NOAA's OR&R [290] along with many sources from U.S. data portals [289], [409] specific restrictions may apply for use and reproduction outside the U.S.

3.6. Conclusions and Observations

The last decade has led to the full recognition of the crucial role played by this new age of decision-support information systems in transportation; the environment; hydrology; meteorology; oceanography; emergency, hazard and disaster management; defense and intelligence; public safety and law enforcement *etc*. When developing such a demanding information system application or research model, obtaining the sufficient amount of the appropriate real-world data, to make the application or the model work effectively, is a requirement of crucial importance. This section aims to provide comprehensive insights into the exploitation of maritime geospatial datasets available to the public and highlights the fact that integrating these datasets from the available online open sources will improve advantageously the building of efficient maritime GIS, while combining them with other restricted and non-free-of-charge data is also made possible.

To the best of the authors' knowledge this study represents the first endeavor to compile a comprehensive survey of carefully selected official online sources, which have been classified under several distinct categories and which, can provide an up-to-date thesaurus of reliable high-precision real-world maritime geospatial data on the international global level. The section also stresses the need to pay due attention to the legal binds that must be taken into account before downloading and using data which are available free-of-charge.

Moreover, this study allowed us to identify open research topics in query processing, big data management and applied machine learning. Focused on skyline queries, we identified that the temporal parameter, which is quite important in data analytics, is not considered in the query process and thus, we focused our research on temporal skyline queries. Moreover, the volume of most of the datasets is large and simple, non-distributed approaches struggle to perform. Based on this we managed to compute the skyline and the even more resource demanding reverse skyline query over one of the largest datasets identified (OpenstreetMap All_nodes dataset [383]) using an index-based approach over Hadoop, named SpatialHadoop. Finally, one of the most common issues identified is the lack of labeled data which is an especially hard process to perform in high volume datasets. Based on this we used the properties of skyline queries to build a classifier that efficiently works in big data environments.

4. SKYLINE QUERIES OVER SPATIO-TEMPORAL DATA

The computation of skyline has been studied across a wide range of environments and types of data. Through our previous study we identified that the notion of time has a great importance in data analytics and query processing. On this scope, a field of study that has remained unexplored in the context of skyline query computation and which would greatly benefit from a study is skyline queries considering the time domain. In many cases time is a critical variable that in many cases is omitted. Specific time intervals may alter the results or even produce different insights. In this study we present that time and its intervals has a great impact on skyline queries since an optimization approach considering the whole-time domain may not be efficient or practical.

4.1. Introduction

In recent years, the skyline query [8] has received a considerable amount of attention because of its ability to highlight in an efficient way the most eligible subset of a set of objects on the basis of a bunch of user-defined criteria. In the following example it is assumed that a traveler does a search for a hotel room. The price of a room is expected to increase as the distance of the hotel from the city center decreases. On the basis of the dataset of Table 26, and by taking into account the first two columns as the primary decision criteria, the potential optimal selection for the user's preferences would be {*a*, *b*, *d*} as presented in Figure 32**.**

Hotel	Price (€)	Distance from the city's center (Km)	Operation Season (months of the year)	
			Start	End
a	15	1,200	1	10
b	25	550	4	8
c	45	1,000	6	10
d	95	200	5	7
e	103	350	3	10
f	147	275	6	7
g	80	850	5	7
h	70	670	6	8
i	65	1,400	5	10
j	83	1,300	7	12

TABLE 26: A DATASET WITH TEMPORAL PARAMETERS.

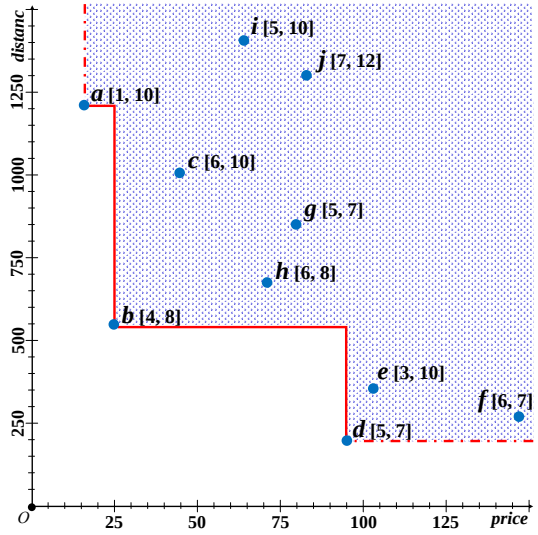


FIGURE 32: THE SKYLINE OF THE DATASET.

Nevertheless, the skyline operator has not been optimized yet to handle temporal data. For example, it is not known how the skyline query can handle the case in which the traveler plans to book a room months ahead by comparing hotels that do not operate all year around. Hence the focus of this study is the efficient temporal extension of the skyline query for temporal data. In this class of data the time period of interest needs to be added as an additional constraint to be evaluated together with the decision criteria of the traditional skyline query. On this basis, the optimal selection that will cover the desired scenario on the dataset of the example of Figure 32 for hotels operating in the 4th month of the year would be the set of hotels {*a*, *b*, *e*} which differs from the set retrieved by applying the traditional skyline query without considering the time domain.

The extension of the skyline query for temporal data aims to demonstrate how the strategy for calculating the traditional skyline query is affected when also considering the time factor. Algorithms for processing modified versions of the static, dynamic, and reverse skyline queries for temporal data are proposed. The key contributions of this study are:

- a new dominant method for evaluating temporal data using the skyline operator,
- algorithms for computing temporal skylines and two of its well-known variants,

- an extensive experimentation on the efficiency of the above algorithms for optimizing the skyline query processing to handle temporal data.

4.2. Problem Formulation

The study involves the extensions of the static, dynamic and reverse skyline queries for the handling of temporal data. It will focus in one dimension of time, which can be either the transaction or the valid time and will comment on the straightforward extension of the proposed solution to handle both time dimensions. The following definitions will make clear the main angles of this study.

Definition 1 - Temporal dominance: Given a time-varying point dataset P in a d-dimensional space D and a point p (p1, p2, ..., pd) $\in$ P with validity in the time interval tp, the point p temporally dominates in the time interval t another point r (r1, r2, ..., rd) $\in$ P with validity in the time interval tr, denoted as p $\prec_t$ r, if and only if t is the non-null intersection between the time intervals tp and tr and $\forall$ i $\in$ {1, ..., d} we have pi $\leq$ ri and $\exists$ j $\in$ {1, ..., d}: pj < rj.

Definition 2 - Temporal Skyline Query: Given a time-varying point dataset P in a d-dimensional space D, the temporal skyline query in the time interval t_s retrieves the set of time-varying points $SL_{ts}(P) \subseteq P$ which are not temporally dominated by any other point in P in any non-null time interval $t \subseteq t_s$, that is, $SL_{ts}(P)$ = {(p, t), where p $\in$ P | $\nexists$ r $\in$ P: r $\prec_t$ p, where $t \subseteq t_s$ is the time interval in which p dominates r}. $SL_{ts}(P)$ is called the temporal skyline of P in the time interval t_s.

The **Table 26** presented the temporal database of ten data tuples represented in **Figure 32** by time-varying points *P* = {*a*, *b*, ..., *j*} in the two-dimensional space. Some data points in the figure temporally dominate others: Point *b*, temporally dominates point *c* in the time interval [6, 8]. The temporal skyline of *P* in the time interval [3, 8] is the set $SL_{[3, 8]}(P)$ = {(*a*, [3, 8]), (*b*, [4, 8]), (*d*, [5, 7]), (*e*, [3, 4]), (*e*, [8, 8])}. Note that point *e* is part of the temporal skyline of *P* in two different time intervals.

Definition 3 - Dynamic Temporal Dominance: Given a time-varying point dataset P in a d-dimensional space D and a reference query point q (q1, q2, ..., qd) $\in$ D with validity in the time interval tq, a point p (p1, p2, ..., pd) $\in$ P with validity in the time interval tp dynamically temporally dominates another point r (r1, r2, ..., rd) $\in$ P with validity in the time interval tr with regard to q in the non-null time interval t, denoted as p $\prec_{(q, t)}$ r, if and only if t is the non-null intersection between the time intervals tp, tr and tq, and $\forall$ i $\in$ {1, ..., d} we have |qi − pi| $\leq$ |qi − ri| and $\exists$ j $\in$ {1, ..., d}: |qj − pj| < |qj − rj|.

Definition 4 - Dynamic Temporal Skyline Query: Given a time-varying point dataset P in a d-dimensional space D and a reference query point q (q1, q2, ..., qd) $\in$ D with validity in the time interval tq, the dynamic temporal skyline query of P with regard to q in the time interval tq retrieves the set SL(q, tq)(P) of points in P which are not dynamically temporally dominated by any other point in P in any non-null time interval $t \subseteq tq$, that is, SL(q, tq)(P) = {(p, t), where $t \subseteq tq$ and p $\in$ P | $\nexists$ r $\in$ P: r $\prec_{(q, t)}$ p}. SL(q, tq)(P) is called the dynamic temporal skyline of P with regard to q in the time interval tq.

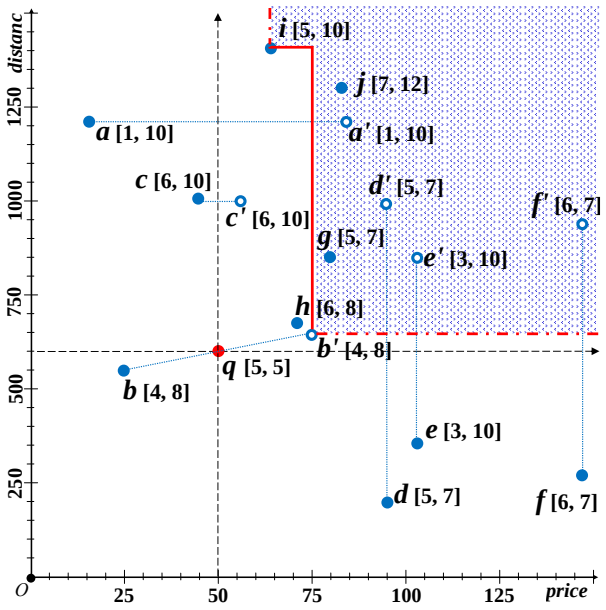


FIGURE 33: THE DYNAMIC TEMPORAL SKYLINE OF THE DATASET OF Table 26 WITH REGARD TO A QUERY POINT Q IN THE TIME INSTANT 5.

In

Figure 33 every database point $p(p_x, p_y)$ in the original 2-dimensional space of Figure 32 is transformed into a point $p'(|q_x - p_x|, |q_y - p_y|)$ in a new 2-dimensional space, the origin in which is the query point q (50, 600) with validity in the time interval [5, 5], i.e. in the time instant 5. The dynamic temporal skyline of P with regard to q in the time interval [5, 5] consists of the set $SL_{(q, [5, 5])}(P)$ = {(*b*, [5, 5]), (*i*, [5, 5])}, whereas the dynamic temporal skyline of P with regard to the same query point q in the time interval [5, 7] consists of the set $SL_{(q, [5, 7])}(P)$ = {(*b*, [5, 7]), (*c*, [6, 7]), (*h*, [6, 7]), (*i*, [5, 5])}. Again, it is possible for a data point to be part of the dynamic temporal skyline of a dataset in more than one subinterval.

Definition 5 - Reverse Temporal Skyline Query: Given a time-varying point dataset P in a d-dimensional space D and a reference query point q (q1, q2, ..., qd) $\in$ D with validity in the time interval tq, the reverse temporal skyline query of P with regard to q in the time interval tq retrieves the set RSL(q, tq)(P) of points in P which take q as one of their dynamic temporal skyline points in the non-null time-interval t $\subseteq$ tq. This means that a point p $\in$ P with validity in the time interval tp belongs to the set RSL(q, tq)(P) and therefore is a reverse temporal skyline of q in the time-interval t, if there does not exist any other point r $\in$ P with validity in the time interval tr such that (1) t is the non-null intersection between the time intervals tp, tr and tq, (2) $\forall$ i $\in$ {1, ..., d}: |ri − pi| $\leq$ |qi − pi| and (3) $\exists$ j $\in$ {1, ..., d}: |rj − pj| < |qi − pj|. RSL(q, tq)(P) is called the reverse temporal skyline of P with regard to q in the time interval tq.

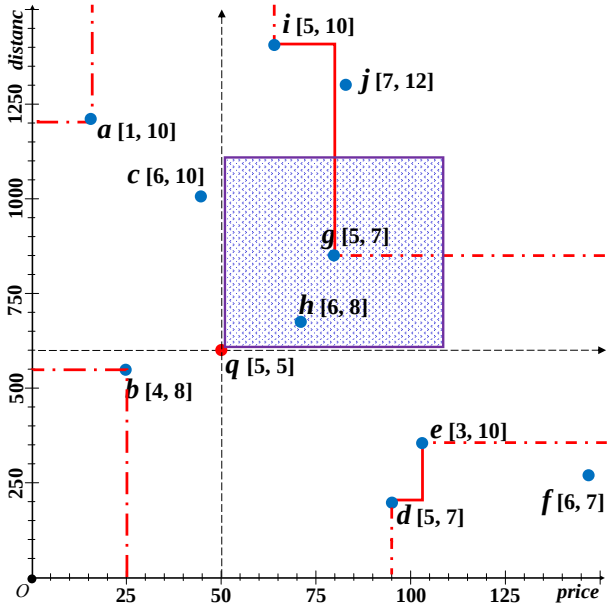


FIGURE 34: THE REVERSE TEMPORAL SKYLINE OF THE DATASET OF TABLE 26 WITH REGARD TO A QUERY POINT Q IN THE TIME INSTANT 5.

In the example of Figure 34, the reverse temporal skyline of *P* with regard to query point *q* (50, 600) in the time interval [5, 5], i.e. in the time instant 5, consists of the set $RSL_{(q,\,[5,\,5])}(P)$ = {(*a*, [5, 5]), (*b*, [5, 5]), (*g*, [5, 5]), (*i*, [5, 5])}. For instance, since the dynamic temporal skyline of data point *g* in the time instant 5 contains the query point *q* (i.e., this holds because no any other data point exists in the grey range of Figure 34 in the time instant 5), *g* is a reverse skyline point of *q* in that time instant.

4.3. Skyline Query Processing Over Temporal Data

This section will formally present the algorithms for implementing efficiently the three new extensions of the skyline query in the temporal databases domain, i.*e*., the static, dynamic and reverse temporal skyline queries.

4.3.1. The Temporal Skyline Query

The algorithm for computing the temporal skyline of a time-varying point dataset is an extension of the original BBS algorithm [3] for traditional (non-temporal) data. Since BBS uses a typical data-partitioning method, such as the R-tree, to serve as the backbone indexing method, in this paper the 3D R-tree access method [79] is considered to be the best choice for maintaining the temporal data. The reason for this choice is that the description of the 3D R-tree differs only slightly from that of the traditional R-tree in respect of its ability to store transaction and/or valid time data as extra data dimensions in the tree. Another reason for selecting the 3D R-tree is that it is accompanied by a simple implementation and requires the fewest possible modifications to the built-in functionalities of modern database management systems as compared to its competitors in the temporal databases domain. The 3D R-tree can straightforwardly support as many user-defined data dimensions as required for any skyline query processing application as compared to most of its temporal indices competitors, which can support only a single dimension for the key of the data tuples, plus of course one or two time dimensions.

```
Algorithm 1: The temporal skyline query ()
Input:       A dataset P, indexed using a 3D R-tree
             and a requested time interval t_s.
Output:      The temporal skyline SL_ts(P).
-------------------------------------------------------
       SL_ts = H = ∅; //H is a heap
1:     FOR every 3D R-tree root entry e with validity
         in the time interval t_e DO
2:       IF t_e ∩ t_s ≠ ∅ THEN insert (e, t_e ∩ t_s) into H;
3:     WHILE H is not empty DO
```

```
4.      Remove top entry (e, t_e) of H;
5:      FOR every interval t ⊆ t_e in which e is not
          temporally dominated by any point in SL_ts DO
6:        IF e is an intermediate entry THEN
7:          FOR every child ee of e, with validity
              in the interval t_ee with t ∩ t_ee ≠ ∅ DO
8:            FOR every time interval t' ∈ t ∩ t_ee
                in which ee is not temporally
                dominated by any point in SL DO
9:              Insert (ee, t') into H;
10:       ELSE // e is a data point
11:         Insert (e, t) into SL_ts;
        RETURN SL_ts;
```

ALGORITHM 1: THE TEMPORAL SKYLINE QUERY.

The pseudo code of the algorithm for computing the temporal skyline is illustrated in **Algorithm 1**. The proposed algorithm makes temporal dominance checks by considering independently the time dimension. The point dataset of **Figure 32** will be used, organized in the four MBRs R_1, R_2, R_3 and R_4 that are illustrated in **Figure 35**. For simplicity, it will be assumed that the root node of the 3D R-tree holds only these four MBRs. The distances are computed according to L_1 norm, i.e., the *mindist* of a data point to the origin point O of the data space is equal to the sum of its coordinates while the corresponding *mindist* of an MBR equals the *mindist* of its lower-left corner point.

The requested time interval to compute the temporal skyline is assumed to be the t_s = [3, 8]. The algorithm in Lines 1-2 starts from the 3D R-tree root node and inserts all its entries with time validity overlapping the requested time interval in a heap H, in the form {(R_2, [3, 8]), (R_1, [3, 8]), (R_3, [5, 8]), (R_4, [3, 8])}, sorted according to the MBRs' *mindist*. Then, by executing the Lines 4-9 of the algorithm, the MBR entry (R_2, [3, 8]) with the minimum *mindist* will be replaced in the heap by its data entries, in the form: (*d*, [5, 7]), (*f*, [6, 7]), and (*e*, [3, 8]).

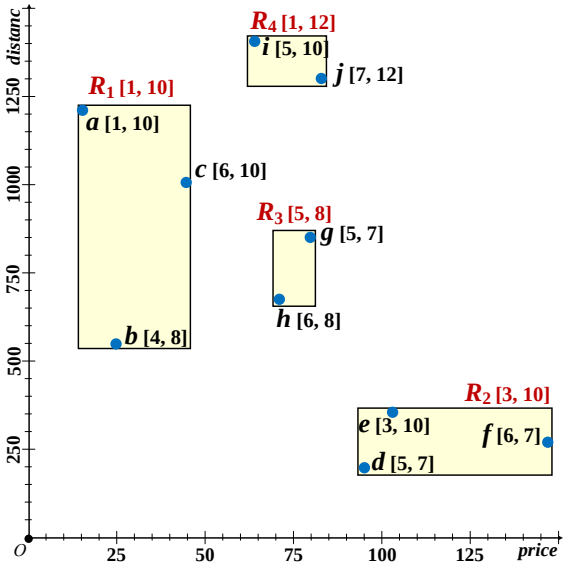


FIGURE 35: THE DATASET OF Figure 32 ORGANIZED IN FOUR MBRS.

The next entry to be extracted from the heap according to **Table 27** is (d, [5, 7]), which, according to Line 11 of the algorithm, is inserted into the temporal skyline list. The next entry to be extracted from the heap is (f, [6, 7]) for which, in Line 5 of the algorithm, it is discovered that it is temporally dominated in every time instant in the interval [6, 7] by entry (d, [5, 7]) of the temporal skyline. The next entry to be extracted from the heap is (e, [3, 8]) for which, in Line 5 of the algorithm, it is discovered that it is not temporally dominated in the time intervals [3, 4] and [8, 8], therefore the corresponding entries (e, [3, 4]) and (e, [8, 8]) are inserted in the temporal skyline. The MBR R1 is then expanded and, as **Table 27** shows, its contents are inserted in the heap. Then entry (b, [4, 8]) of the heap is inserted in the temporal skyline. Subsequently, the entry of the MBR R3 is extracted from the heap and it is found that it is temporally dominated by the data point b in every

time instant in the interval [5, 8] in which the MBR is valid. Finally, after processing some more data entries, the MBR R4 is extracted from the heap, which, however, is temporally dominated in the entire requested time interval [3, 8] of the query.

action	***H* content**	***SL*$_{[3, 8]}$ () content**
expand root in [3, 8]	(R_2, [3, 8]), (R_1, [3, 8]), (R_3, [5, 8]), (R_4, [3, 8])	–
expand R_2 in [3, 8]	(d, [5, 7]), ~~(f, [6, 7]),~~ (e, [3, 8]), (R_1, [3, 8]), (R_3, [5, 8]), (R_4, [3, 8])	(d, [5, 7]), (e, [3, 4]), (e, [8, 8])
expand R_1 in [3, 8]	(b, [4, 8]), ~~(R_3, [5, 8]),~~ ~~(c, [6, 8]),~~ (a, [3, 8]), ~~(R_4, [3, 8])~~	(d, [5, 7]), (e, [3, 4]), (e, [8, 8]), (b, [4, 8]), (a, [3, 8])

TABLE 27: PROCESSING STEPS OF THE EXAMPLE EXECUTION OF ALGORITHM 1

The correctness of the proposed algorithm is straightforwardly inherited from the corresponding correctness [3] of the BBS algorithm for traditional (non-temporal) data. This means that every data point added into the temporal skyline during the execution of the algorithm is guaranteed to be a final temporal skyline point for the time interval under consideration. Also, every data point in the 3D R-tree will be examined by the algorithm, unless one of its ancestor nodes has been pruned for the whole time interval of the validity of the data point. The proposed algorithm is also progressive, it provides neither false misses nor false hits and it is able to allow the user to determine the order in which skyline points will be returned.

In the case of bi-temporal data, the algorithm can perform temporal dominance checks by considering every time dimension independently, which means that a data point belongs to the temporal skyline only if it is not temporally dominated by any other point in the dataset in both the valid and transaction time dimensions.

4.3.2. The Dynamic Temporal Skyline Query

While the static temporal skyline evaluates the data objects on the basis of the minimum (or maximum) values of their coordinates, the dynamic temporal skyline evaluates the data objects in respect of a customer's given preference point q (q_1, q_2, …, q_d) in a specified time interval t_q (a hotel at 50 euros, at a 600 meters from the city center, the following April). Therefore, the dynamic temporal skyline query with regard to q in the time interval t_q, for every data point p (p_1, p_2, …, p_d) with validity in the time interval t_p which overlaps t_q, specifies d functions of the form $\forall\, i \in \{1, ..., d\}$: $f_i = |q_i - p_i|$, and the goal is to return the static temporal skyline of P in the time interval t_q, in the transformed/dynamic workspace which has q as its point of origin and the coordinates of every object p in every dimension are defined by the functions fi.

The **Algorithm 1** is applicable to dynamic temporal skylines by storing in the heap the entries according to their *mindist* in the dynamic workspace. See [3] for more details. The main modifications that are needed so that **Algorithm 1** can process the dynamic temporal skyline query is the replacement of the temporal dominance checks in Lines 5 and 8 by dynamic temporal dominance checks, as they are set out in Definition 3.

4.3.3. The Reverse Temporal Skyline Query

As with the dynamic temporal skyline, the reverse temporal skyline evaluates the data objects with regard to a given query point q on a specified time interval t_q. However, the main difference between these two queries is that the dynamic temporal skyline query can be seen as a query from the customer's perspective whereas the reverse temporal skyline can be seen as a query from the company's perspective. Therefore in the reverse temporal skyline case the customer's preferences are represented by data points in the workspace and the query point q is set by the company to determine the effectiveness of a particular product (which customers would be interested in a hotel room at 50 euros, at 600 meters from the city center, between October and May?).

Four different algorithms for processing the reverse skyline query for traditional (non-temporal) data are proposed in [14] and [430], with the Branch and Bound Reverse Skyline (BBRS) algorithm [14] to be the one selected to serve as a backbone algorithm for extension in order to

support the reverse temporal skyline. The BBRS algorithm is chosen for the simplicity of its implementation and its ability to run without the need to preprocess the dynamic skyline of every point in the dataset. The drawback of the BBRS in comparison to its three competitors is that it requires that the index be traversed once for every candidate reverse skyline point that is found in the final filtering step of the algorithm. This can be easily overcome by ensuring that the algorithm is accompanied by a buffer to hold the most frequently- or the least recently- used nodes of the index in memory for faster potential future usage.

```
Algorithm 2: The reverse temporal skyline query ()
Input:       A dataset P, indexed using a 3D R-tree,
             a query point q (q1, q2, …, qd) and a
             time interval tq.
Output:      The reverse temporal skyline RSL(q, tq) (P)
      RSL = H = ∅; //H is a heap
1:    FOR every 3D R-tree root entry e with validity
        in the time interval te DO
2:      IF te ∩ ts ≠ ∅ THEN insert (e, te ∩ ts) into
3:    H;
4:    WHILE H is not empty DO
5:      Remove top entry (e, te) of H;
        FOR every interval t ⊆ tq ∩ te in which e is
          not globally temporally dominated by any
6:        point in RSL DO
7:        IF e is an intermediate entry THEN
            FOR every child ee of e, with validity
8:            in the interval tee with t ∩ tee ≠ ∅ DO
              FOR every time interval t' ∈ t ∩ tee in
                which ee is not globally temporally
9:              dominated by any point in RSL DO
10:             Insert (ee, t') into H;
11:       ELSE // e is a data point
12:         Execute a range query based on e, q, t;
            IF the range query is empty in any time
13:           interval t' ⊆ t THEN
14:           Insert (e, t') into RSL;
      RETURN RSL;
```

ALGORITHM 2: THE REVERSE TEMPORAL SKYLINE QUERY.

The pseudo code of the proposed algorithm is illustrated in **Algorithm 2**. The algorithm in Lines 5 and 8 makes global temporal dominance checks according to the following definition.

Definition 6: Global Temporal Dominance: Given a time-varying point dataset P in a d-dimensional space D and a reference query point q (q_1, q_2, ..., q_d) $\in D$ with validity in the time interval t_q, a point p (p_1, p_2, ..., p_d) $\in P$ with validity in the time interval t_p *globally temporally dominates* another point r (r_1, r_2, ..., r_d) $\in P$ with validity in the time interval t_r with regard to q in the non-null time interval t if and only if (1) t is the non-null intersection between the time intervals t_p, t_r and t_q, (2) $\forall\, i \in \{1, ..., d\}$: $(p_i - q_i)(r_i - q_i) > 0$, (3) $\forall\, i \in \{1, ..., d\}$: $|p_i - q_i| \leq |r_i - q_i|$, and, (4) $\exists\, j \in \{1, ..., d\}$: $|p_j - q_j| < |r_i - q_j|$.

On the basis of the definition and of the example of **Figure 34**, it can be said that the point g globally temporally dominates the point j in the time interval [5, 5].

The global temporal dominance checks in the algorithm help with the pruning of intermediate index nodes (and data points) which cannot store (or be, respectively) reverse temporal skyline points. The first ten lines of the algorithm are executed in a similar manner to **Algorithm 1** for the (static) temporal skyline. However, for every point e with validity t_e overlapping the time interval t_q of the given query q that is not globally temporally dominated in a time interval $t \subseteq t_e \cap t_q$, a further examination is required. This examination is performed in Lines 11-12 of the algorithm by issuing a range query, with e being in the centre of the range window and q in its corner, similarly to that illustrated in grey around the data point g in **Figure 34**. As **[14]** shows for the case of non-temporal data, if this range query returns no data point for a time interval $t' \subseteq t$, then e is a reverse temporal skyline point with regard to q in the interval t'. Therefore, in this case in Line 13 of the algorithm the tuple (e, t') is inserted into the $RSL(q, t_q)$ list.

In the reverse skyline query of Figure 34 with regard query point q (50, 600) and time instant 5, the data entries which are valid and not globally temporally dominated by any other point in this instant are a, i, h, b, d, and e. However, after performing the range query checks of Lines 11-12 of the algorithm, only the first four of these data points is found to belong to the reverse temporal skyline of the dataset with regard to q in the time instant 5.

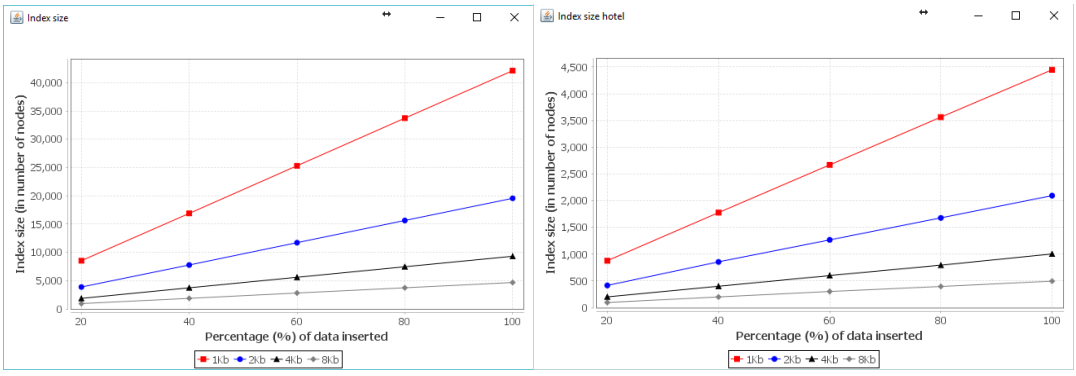
The handling of bi-temporal data can be treated in an analogous manner to the (static and dynamic) temporal skyline query, i.e., by performing global temporal dominance checks in every time dimension independently.

4.4. Experimental Study

The proposed query algorithms were implemented in Java (JDK version 8). The 3D R-tree implementation is based on the R*-tree implementation in Java that can be downloaded from the ChoroChronos portal[3]. The workstation that was used for evaluation, was equipped with Intel I7 6GB RAM running the Windows 8.1 Professional 64-bit OS. The Java Virtual Machine Heap was set to its default values.

The experiments have been conducted using two datasets. The first is a synthetic dataset which is constructed by 1,000,000 uniformly distributed time-varying two-dimensional points with a uniformly distributed time interval validity of maximum 20% of the lifespan of the scenario, which is 1,000 time instants. The second is the real-life Major Hotel Chain dataset [431] having 147,029 bookings collected from five U.S. properties of a major hotel chain. In order to construct a two-dimensional point for every booking record, the Nightly_Rate column was considered plus a uniformly distributed artificial column with values between 0 and 100, which could for example represent the customer's rating score for the service provided by the hotel. The validity time interval of every booking is constituted by the combination of the columns Check_In_Date and Check_Out_Date as they are given by the data provider.

Every experiment has been repeated 10 times and the average value of every measured parameter has been calculated. As with regard to the dynamic and the reverse temporal skyline queries, at every run a different randomly selected query point has been used. In the following, unless otherwise stated, the findings of the performance investigation of the proposed query processing algorithms are qualitatively comparable, whether the synthetic or the real data are used, therefore in some cases only half of them (*i.e.*, either with the synthetic or with the real data) is depicted in the paper. Finally, four different values are considered in the experiments for the file system page size, *i.e.*, 1K, 2K, 4K and 8K. The 3D R-tree node size is set to be equal to the page size.



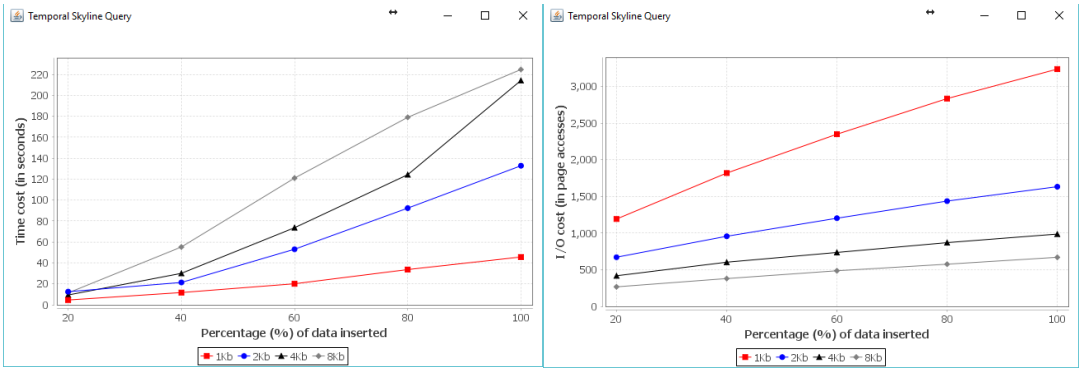
(a) (b)

FIGURE 36: THE 3D R-TREE INDEX SIZE IN A NUMBER OF NODES, (A) FOR THE SYNTHETIC DATASET, AND (B) FOR THE REAL DATASET.

The first two graphs in Figure 36 show the size of the 3D R-tree index for the synthetic (on the left) and for the real (on the right) datasets. The index size has been measured every 20% percent

[3] http://chorochronos.datastories.org

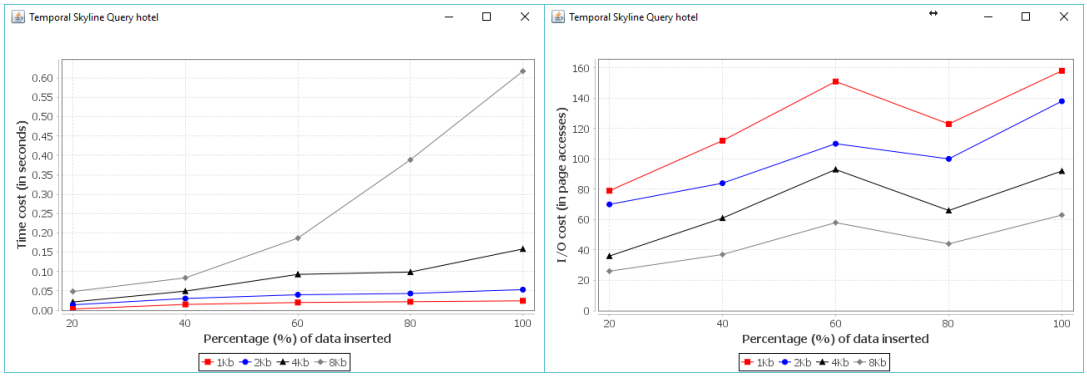
of the data being inserted. These results will help the measurement of the % percentage of the index that is accessed when processing every query in the graphs that will be follow.



(a) (b)

FIGURE 37: (A) THE TIME COST, AND (B) THE I/O COST, IN BOTH CASES FOR EXECUTING THE TEMPORAL SKYLINE QUERY ALGORITHM FOR THE SYNTHETIC DATASET.

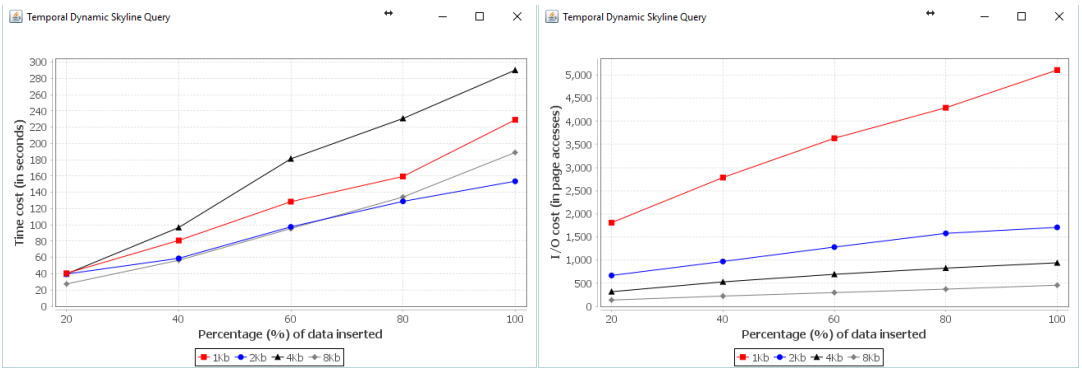
The next two graphs in Figure 37 illustrate the time cost in seconds (on the left) and the I/O cost in page accesses (on the right) for answering the temporal skyline query using the synthetic dataset. The query is executed every 20% percent of the data being inserted, and in every case for a time interval that is equal to the lifespan of the scenario, thus the skyline is computed for every time instant in the lifetime of the scene. By comparing the I/O cost to the corresponding index size that is shown in Figure 36(a), it is concluded that the temporal skyline algorithm accesses about the 8% to 23% of the index. This cost is justified by the large number of the 1,000 time instants for which the skyline is calculated with only a single tree traversal using the Algorithm 1.



(a) (b)

FIGURE 38: (A) THE TIME COST, AND (B) THE I/O COST, IN BOTH CASES FOR EXECUTING THE TEMPORAL SKYLINE QUERY ALGORITHM FOR THE REAL DATASET.

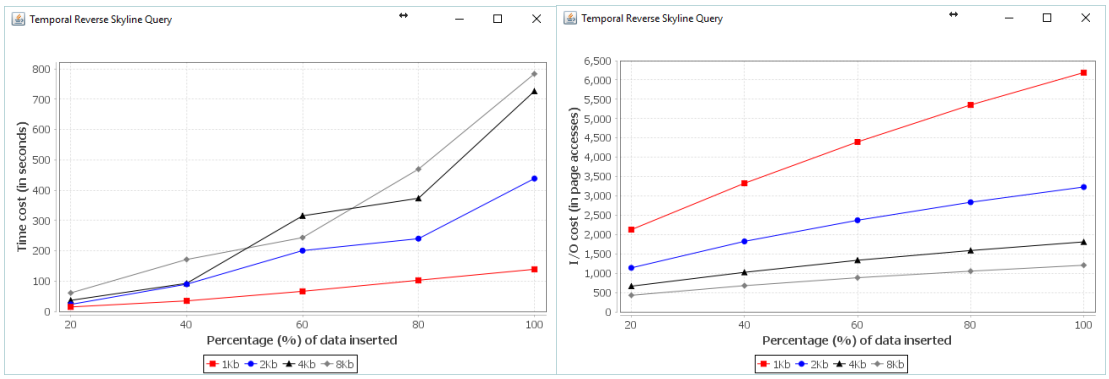
The next graphs in Figure 38 illustrate the time (on the left) and the I/O (on the right) efficiency of the temporal skyline query algorithm for the real-life dataset. The query is again executed for computing the skyline for every time instant in the lifetime of the scene. By comparing the I/O cost to the corresponding index size that is shown in Figure 36(b), it is found that the temporal skyline algorithm accesses about the 3% to 19% of the index. This cost is also justified by the large number of the time instants for which the skyline is calculated with only a single tree traversal. The difference in cost, between the synthetic and the real dataset, is due to the distribution of time instances that allow more nodes to be dominated in the real dataset.



(a) (b)

FIGURE 39: THE TIME COST, AND (B) THE I/O COST, FOR EXECUTING THE DYNAMIC TEMPORAL SKYLINE QUERY ALGORITHM FOR THE SYNTHETIC DATASET.

The next graphs in **Figure 39** show the time cost (on the left) and the I/O cost (on the right) for the execution of the dynamic temporal skyline query algorithm using the synthetic dataset. The I/O cost for the execution of the algorithm for this "dynamic" query is larger than in the case of the "static" query since it must access a larger portion of the dataset.



(a) (b)

FIGURE 40: (A) THE TIME COST, AND (B) THE I/O COST, FOR EXECUTING THE REVERSE TEMPORAL SKYLINE QUERY ALGORITHM FOR THE SYNTHETIC DATASET.

The graphs of the last experiment in **Figure 40** show the time cost (on the left) and the I/O cost (on the right) for the execution of the reverse temporal skyline query algorithm using the synthetic dataset. The results show an expected difference in I/O cost and much higher time cost performance in relation to the dynamic temporal skyline even if both of these queries access quite similar parts of the data space (thus of the index) when the same reference query points *q* are used. This happens because the reverse temporal skyline has an additional overhead due to the empty/Boolean range query. In every case the distribution and the length of the time intervals make a large impact on the execution time and the I/O cost. Two extreme examples will be the case where all time intervals in the dataset to be distinct and non-overlapping and the other to be identical. In the first case the algorithm must traverse the entire tree and return all its points and in the second case the algorithm becomes the initial simple variant of each query. These problems need a more sophisticated solution and multiple refinement mechanisms in order to compute the time variant efficiently. In every case the computation of the temporal variant in the whole temporal dimension will be an expensive task.

4.5. Conclusions and Future Work

The skyline query is a decision support mechanism which, in essence, retrieves the so-called value-for-money options of a dataset by identifying the objects which present the optimal combination of the characteristics of the dataset. This study is the first to take the time factor into consideration and it optimizes the skyline operator as well as two of the most well-known of its

variants, i.e. the dynamic and the reverse skyline operators, to handle temporal data. It is anticipated that the results of this research pave the way for the construction of other solutions for processing efficiently skyline-based queries for a variety of temporal and bi-temporal data applications.

Future plans are to investigate the impact of the backbone temporal indexing method on the performance of the queries execution cost, by comparing the performance of several appropriate indexing methods or by suggesting new efficient ones for the problem into consideration. Additional plans are to investigate the impact on the performance of the queries of the existence of many objects with relatively small or large time interval lifespans. Another interesting issue for future research is the introduction of efficient algorithms for processing extensions of other skyline query variants that can be also applied to temporal data, such as for example, the support of the so called why-not reverse skyline query **[55].** The aim of this temporal query will be to make a product (time-varying query point) interesting to a customer (time-varying why-not point) by modifying the product's features (query attributes) and/or the customer preferences.

5. SKYLINE QUERIES OVER SPATIALHADOOP

Through our first study on data sources we identified that many of the datasets fall into the big data domain in terms of volume, variety, velocity and veracity. The vast amount of data and the need to process them produced new technologies like the Hadoop ecosystem. Based on this existing querying approaches should be studied over those new environments in order to see the insights that they produce. In this chapter, we study the problem of skyline and reverse skyline computation using SpatialHadoop, an extension of Hadoop that enhances its capabilities with spatial awareness. The exploitation of spatial indexing structures and the spatial properties of data can exploit MapReduce-based methods by reducing the reading, writing, computational and communicational overhead. Through our study, we propose two methods for skyline and reverse skyline computation, which operates in the spatial aware environment that SpatialHadoop provides. This environment allows for performing filtering on the initial dataset to retrieve an answer efficiently by using existing state-of-the-art indexing approaches. The proposed algorithms make use of the full capabilities of the indexing mechanisms provided by the SpatialHadoop and have been tested against large-scale datasets including a real-life, large-scale OpenStreetMap dataset. To the best of our knowledge, this is the first work that studies reverse skyline over SpatialHadoop.

5.1. Introduction

The trend of *Big Data* was one of the most discussed research topics in the past years. Nowadays, this trend has managed to be a well-established technology that has changed the way many industries operate. Beyond that, *Big Data* is the driving force behind many new technologies in *Artificial Intelligence (AI)*, *Internet of Things (IoT)* and *data science*. The *Big Data Era* started with the need for processing the vast amount of structured, semi-structured and unstructured data in a batch or streaming way. Since then, additional aspects were included, based on the scope behind the data such as the value of data in socio-economical terms and the veracity of data in terms of quality and accuracy. In the present time, the term *Big Data* is inseparable from our everyday life since data created in the past two years exceeds all data generated through the whole period of human's digital era. The amount of data generated from a single person's smartphone, activity tracker or shopping habits or even the data generated by a Formula 1 car which is equipped with hundreds of different sensors, is trivial under the trend that even cities are getting smarter and can generate their own data and information.

Storing vast amounts of data can be achieved with *Hadoop's HDFS* **[432]** distributed file system. However, simply storing the data gives no further value to their existence. It is vital to convert data to information in order to acquire knowledge **[433]**. The first step to this direction is to process the stored data in a distributed way with Hadoop's *MapReduce* **[85]**. With *MapReduce* and the use of various methods **[434]**, **[435]** it is possible to retrieve a wide range of information based on the query that is performed. However, even MapReduce finds it difficult to compute all types of queries, since many of them rely on the geometric properties of the dataset like the proximity of points in NN and k-NN queries. At this point, a system like *SpatialHadoop* **[31]** fits best in these types of problems. The *SpatialHadoop* is an extension of *Hadoop* that injects spatial awareness into it. *SpatialHadoop* is not applicable only on spatial data as the idea of *GIS* systems. It is applicable to any kind of data that may have spatial or geometric properties and is designed to harvest the power of those properties to produce methods that will retrieve answers efficiently.

Based on the aforementioned scope, it is shown that *range queries* and *spatial joins* are made much faster by using *SpatialHadoop* **[31]**. Similarly, skyline **[8]** and *reverse skyline queries* **[14]** could benefit from this system, as this study will present. The primary goal in the design of these algorithms is to prune as many points or even partitions as soon as possible and thus minimize the data transfers between *mappers* and *reducers*. To achieve these goals we used *SpatialHadoop* that will help in the pruning process by exploiting the spatial and geometric properties of the dataset.

In summary, the key contributions of this study are the following:

- An alternative approach to the one in [32] for skyline query computation is proposed that is used to enhance SpatialHadoop with reverse skyline queries.

- An algorithm for reverse skyline queries computation is proposed that incorporates a multiple filtering mechanism to allow for the pruning of the dataset as soon as possible.
- Extensive experimentation in large-scale synthetic, real datasets and different environments is performed in order to demonstrate the performance benefits.

5.2. Preliminaries

This section will discuss the basic concepts of *MapReduce,* along with its key components and the mechanisms of *SpatialHadoop* used in different implementations of the *skyline* and *reverse skyline* query computation.

5.2.1. MapReduce

The MapReduce framework can perform processing of massive amount of data in parallel, over multi-node clusters efficiently. The data that the framework processes can be stored in the HDFS. The framework primarily consists of two main phases, named *map* and *reduce*.

In the *MapReduce* framework, every *value* has an associated *key*. These *keys* allow identifying related *values*. The *map* phase processes *splits* of the input dataset that contains the *key-value* pairs and can output a different number of altered *key-value* pairs. **Figure 41** describes the workflow from reading the dataset/records up to the point where the *map* phase processes the *key-value* pairs. The *reduce* phase retrieves the values from the *map* phase, sorted and grouped by the *keys* and produces a different set of *key-value* pairs. A *job* is the whole mechanism that handles and processes the data and consists of many *map, reduce* and other phases in which the output of one phase becomes the input in another one. Each phase can process data in the following sequence: Mapper→ Combiner→ Partitioner→ Shuffling→ Sorting→ Reducer. **Figure 42** illustrates the complete workflow of a *job***.**

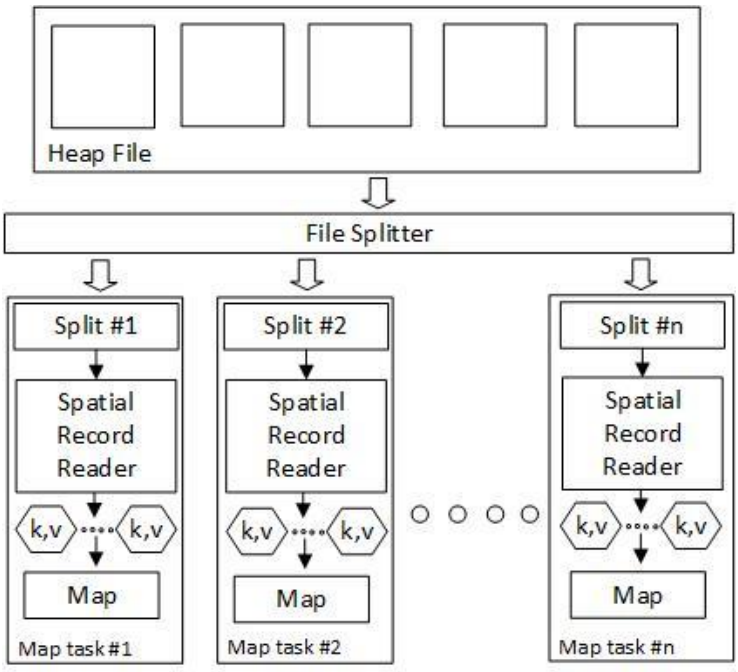


FIGURE 41: HADOOP EXECUTION WORKFLOW AS PRESENTED IN [31].

Two important aspects that researchers faced during *skyline* computations were the early *pruning* of the dataset and the possibility of *sorting* it in order to identify the most promising points in an early stage. The two branches of skyline query computation that proposed *index-based* methods and *sort-based* methods reflect the importance of these two aspects.

The first and major *pruning* mechanism in a *MapReduce job* should be integrated into the *map* phase in order to minimize the network communication cost **[85]** that exists between the *map* and *reduce* phase. The work on **[107]** captures this idea, where the authors build an on-the-fly index

that is used inside the *map* phase to prune a large part of the dataset. Further refinement techniques can be placed in the *combiner* and *reduce* phase.

The other important aspect that *MapReduce* must handle is the *tuple traversal order* and thus the desire of *sorting* the dataset. Since for indexing we used an *R-tree*, both the *BBS* **[3]** *algorithm for skyline* computation *and the BBRS* **[14]** algorithm for *reverse skyline computation* rely on the *Branch & Bound* paradigm. This paradigm dictates to visit the nodes based on the shorter distance from the origin of the axis, which is accomplished with a sorted heap. The same approach can hold with a *Grid* index, which is supported by *SpatialHadoop*, by additionally incorporating the *incomparability* property. In the *MapReduce* framework, the order that the *key-value* pairs appear in a *map* phase is the order in which they are stored in the *HDFS* files or as fetched from another source. Even if the *reduce* phase uses a built-in *sorting* mechanism (*Comparator*) that sorts data to be fed in the *reducer* in ascending order of their *keys*, the only way to have a sorted input in the *map* phase is to perform an in-memory sort operation to the assigned *split*.

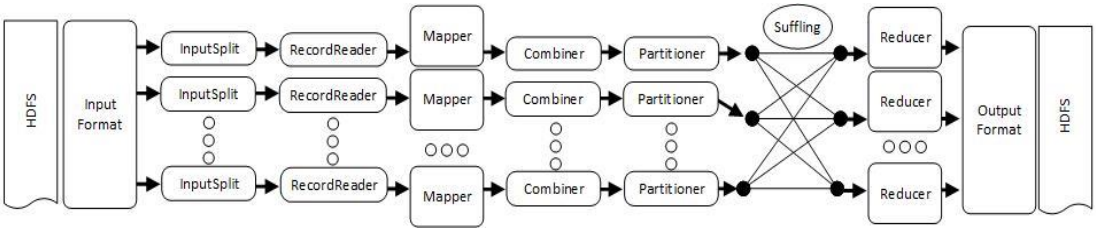


FIGURE 42: MAPREDUCE JOB EXECUTION.

A mechanism that can help in *pruning* and *sorting* of the dataset is the *combiner* phase, which is primarily designed to reduce the communication cost between the *map* and *reduce* phase. A combiner can act as a local *reduce* phase bind to each *map* phase and can be a secondary pruning mechanism by inheriting the *sorting* and *refinements* properties of a *reducer*. A *combiner's class* can be an instance of the *reducer's class or* a custom one if the refinement approach on the *reduce* phase is too early to be performed at this stage of the algorithm. A delicate issue is that this mechanism is designed to be optional and is called when the *map* output becomes too large. Thus, we cannot prove the correctness of an algorithm based on its capabilities. In general, if we perform a *sorting* and *filtering* operation in the *map* phase, a *combiner* may not be useful. On the other hand, the use of a combiner can potentially assist by computing the local skyline on the assigned split to reduce the amount of data transfers and the processing cost in the *reducers*.

With or without the use of a *combiner*, the *key-value* pairs outputted on each *map* phase will be grouped based on their *keys* and each or multiple groups will be directed to specific *reducers* in the process of *shuffling*. With the term *partition,* we describe the set of groups directed to a single *reducer*. The number of *partitions* can be equal to the number of *reducers*. By default, the *key-value* pairs are partitioned based on the hashcode of the *key* modulo the *total number of partitions* divided by the *number of reducers*. An example where a custom *partitioner* could be useful is **[102]** in which the *key-value* pairs could be partitioned using the *MR-Angle* approach. A primary goal will be to distribute the *key-value* pairs evenly to the *reducers*.

An important parameter for the *pruning*, *sorting* and *partitioning* phases is the assigned *key* since all the operations are performed based on that. In the case where the *key* is the *distance* from the *origin point*, most of the *key-value* pairs will have a *unique key*. The combination of this *key* with the built-in sorting mechanism of *reduces* will guarantee the desired *tuple traversal order* based on the optimization criteria of the user. When the cardinality of each final *map* output that will be fed to the *reducer*s is small, a *null key* could be considered. This would result in the initialization of one *reducer* that will process all the data from the *map* phase. A single *reducer* fits in our case since the result set can only be computed using the data from all partitions. One further approach in *key* selection is the use of a *composite key,* which holds two separate *keys*. Using a *composite key* there is the need for a custom *sorting* mechanism, which should override the built-in *sorting* mechanism.

5.2.2. Hadoop and Spatial Awareness

The *MapReduce* framework allows the parallel processing of massive amounts of data that can be stored in a distributed file system, named *HDFS* **[432]** that exists over many independent physical machines. The *MapReduce* inherits parallelism in the computation of different problems, but by design, it does not provide any indexing mechanism. The way that it handles data is by reading the whole or parts of the dataset simultaneously and further processing it sequentially. This may not be the ideal case for many queries, which exploit the geometric and spatial properties of the dataset. A survey that reasons on how the spatial data cope with *MapReduce* is the **[436]** in which authors review the indexing mechanisms for spatial query processing in traditional and *MapReduce*-based approaches. In **[437]** the authors proposed a *Z-Curve* based approach along with a sweeping algorithm and *pending files* to efficiently answer certain queries. Moreover, in **[438]** an *R-tree* construction mechanism in *MapReduce* was proposed. For a complete view of the advantages and disadvantages of *MapReduce* as long as the proposed improvements, the reader can consult **[439]**. Finally, the work on **[440]** studies the subject of query processing in *MapReduce*, which among others, categorizes the weaknesses and the solving techniques as well as techniques for efficient query processing on *MapReduce*.

A mechanism that enhances *MapReduce* with spatial awareness is the *SpatialHadoop* **[31], [441]**. The **[442]** reasons about the four main layers, namely, *language*, *indexing*, *query processing* and *visualization* of *SpatialHadoop* while the work on **[443]** demonstrates the specific approaches to solve *Computational Geometry* problems like *polygon union*, *skyline*, *convex hull*, *farthest pair*, and *closest pair* in *SpatialHadoop*. The algorithm for the *skyline* problem in this work is a simple *divide & conquer* approach. An experimentation on the indexing mechanisms of *SpatialHadoop* can be found in **[444]** while the authors of **[445]** categorize the various systems similar to *SpatialHadoop*.

5.2.3. SpatialHadoop

As already mentioned, the *SpatialHadoop* **[31]** is a *MapReduce* extension designed to add spatial awareness to it. Each *job* in *SpatialHadoop* is an ordinary *MapReduce job* that uses specific *SpatialInputFormat* and *SpatialRecordReader* methods to handle indexed spatial data in the form of multi-level *partitions*. It does not alter the functionality of *MapReduce* but instead, adds methods to process spatial data. Thus, along with the standard *map* and *reduce* functions *SpatialHadoop* contains a *filter function* (*CellFilter*) whose purpose is to prune block of data, before the *map* phase, which will not contribute to the final answer, by examining the minimal bounding rectangles (*MBRs*) of each *partition*. The use of *SpatialHadoop's* built-in queries, deployed on top of the indexed *partitions*, can provide vital information for the optimization of an algorithm in a similar way to the process of sampling. In general, this filtering mechanism reduces the total number of *map tasks* since each block of data that will pass the phase of pruning will invoke a different *map task*. **Figure 43** describes the similar workflow of a *job* in *SpatialHadoop* to the one of *MapReduce* **Figure 41.**

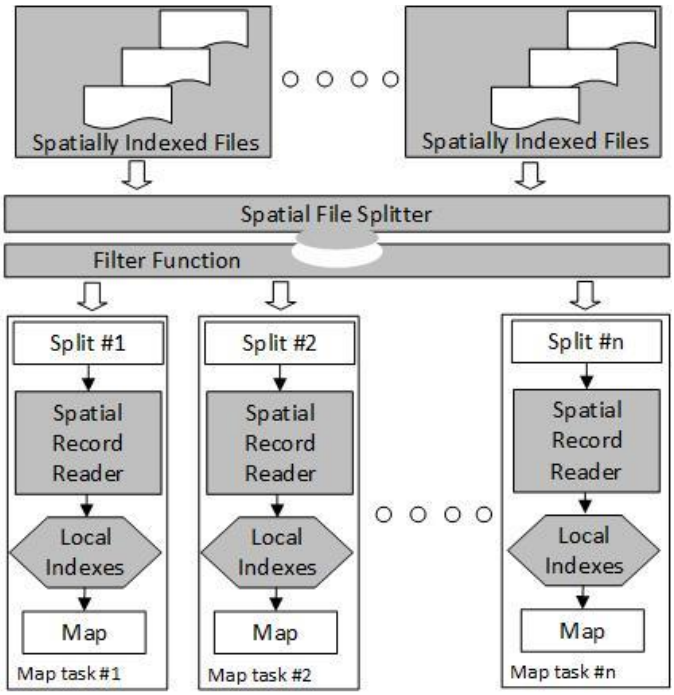


FIGURE 43: SPATIALHADOOP EXECUTION WORKFLOW AS IN [31].

The three different approaches to get *SpatialHadoop's* benefits is by using a *distribution package*, a *portable runnable jar* or a runnable *jar.* Depending on the case, all the required classes and libraries can be loaded on every node startup or installed permanently in the system. Thus, SpatialHadoop is portable to run in *Apache Hadoop*[4], *Cloudera*[5] and *Hortonworks*[6] Hadoop distributions even in their sandboxed version.

The four main layers of *SpatialHadoop* are *language*, *indexing*, *query processing* and *visualization*. The *language* layer has a spatial *MapReduce* language named *Pigeon* [446]. *Pigeon*[7] allows *Pig Latin scripts* to handle spatial data with the use of *ESRI-geometry-API*[8]*.* The *indexing* layer involves many spatial data types such as *points*, *polygons*, *rectangles* and an extensible *shape class* to create your own data types and shapes. Since *SpatialHadoop's* source code[9] is available, multi-dimensional data, greater than *2d,* can be handled by extending or modifying its classes. The indexing mechanisms that are available are the *Grid, R-tree, R+-tree, STR, STR+, Quad-tree, K-d tree, Z-Curve, and Hilbert Curve*. An index can consist of *global* and *local indexes*. The *global index* organizes the partitions across different nodes while *the local index* organizes the data inside each node. For the case of the *R-tree,* its *global index* contains all the *partitions*, while each *local index* contains a local *R-tree* structure with all the data records. With the use of the *SpatialInputFormat*, researchers can access the *global index* to prune partitions and with the SpatialRecordReader they can access, inside the *map* phase, the *local index* to process only the desired records of the whole partition. The *K-d tree* and *Grid* are *local-only indexes*. The *query processing* layers allow running spatial operations using the existing indexing mechanisms. The spatial operation includes *Range, k-Nearest Neighbor (k-NN), Spatial Join, Voronoi Diagram, Delaunay Triangulation, Polygon Union, Convex Hull, Farthest Pair, Closest Pair and Skyline* queries. Every operation is customizable and extendible allowing researchers to develop their own algorithms.

The *visualization* layer [447] allows users to retrieve a *single-level* or *multi-level* image representation of the dataset. A *single-level* image has a standard resolution while the *multi-level*

[4] http://hadoop.apache.org/

[5] https://www.cloudera.com/downloads/quickstart_vms.html

[6] https://hortonworks.com/products/sandbox/

[7] https://github.com/aseldawy/pigeon

[8] https://github.com/Esri/geometry-api-java

[9] https://github.com/aseldawy/spatialhadoop2

image consists of many small image tiles on different regions and at different zoom levels. Researchers can find this feature useful since they can retrieve insights over the dataset with an optical representation.

An important aspect of *SpatialHadoop* is that it can access the *global* and *local indices* in *MapReduce* or in *standalone* mode. In the case of *MapReduce* mode, the *global index* can be accessed prior to the *map* phase and the *local* index inside the *map* phase. The case of *standalone mode* is useful in interactive queries and when the initialization of a new *MapReduce job* is not desired.

5.3. A sort-based Skyline algorithm in SpatialHadoop

The *SpatialHadoop* comes with a built-in algorithm for *skyline query* computation. The algorithm follows the *Divide & Conquer* paradigm in a similar way to the one in **[8],** where the *BNL* algorithm was also proposed. One of the main drawbacks of both algorithms is that they need to process the whole dataset before they are able to identify the first skyline point. This violates the property of *progressiveness* in the set of criteria **[43, 37]** that must be met by all the *skyline* algorithms. Based on these criteria, the state-of-the-art *BBS* **[3]** and *Z-SKY* **[131]** algorithms were proposed.

In the *MapReduce* environment, the authors of **[32]** proposed the algorithms *SKY-FLT* and *SKY-FLT-SORT* for *skyline* computation and compared them to the built-in D&C-based *CG-HADOOP* **[443]** approach. In their work, they showed that both algorithms perform better than *CG-HADOOP*. The *SKY-FLT* algorithm follows a *BNL*-like approach that uses a list of candidate skyline points and the *SKY-FLT-SORT* uses the sorting mechanism of *MapReduce* in order to access the points based on their distance to the origin point in the *reduce* phase. Both algorithms perform an initial pruning process with the use of SpatialHadoop's R-tree. In their work, they showed that even if the *BBS* is the state-of-the-art index-based algorithm that outperforms the original *BNL* algorithm, the *BNL*-like, *SKY-FLT* algorithm performed better in most of the experiments. Additionally, the experimentation made in **[101]** shows that the *MR-BNL* and *MR-SFS,* a variant of the original sort-based *SFS* algorithm, has a similar performance.

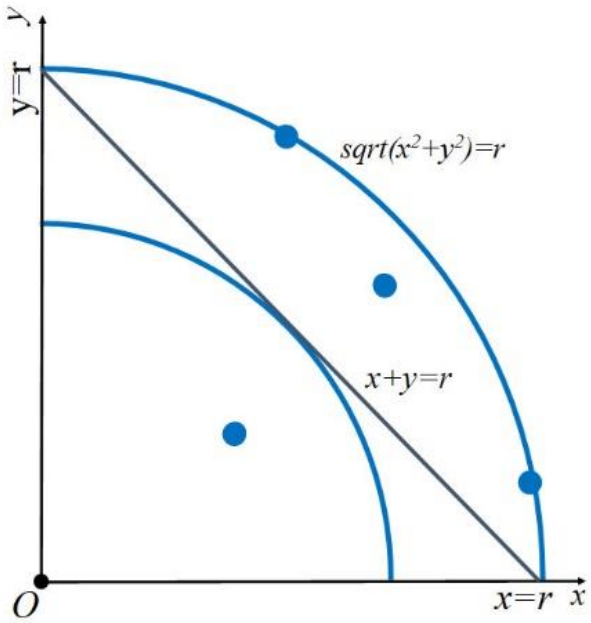


FIGURE 44: POINT ACCESS ORDER WITH MANHATTAN AND EUCLIDIAN DISTANCE MEASURE.

Based on the research in **[32]**, the authors state that their work will not invoke a sorting mechanism inside the *map* and *reduce* phases. We show that with a sort-based model and the pruning power of filtering points, we can achieve better results and the benefit of having *a local skyline* in the output of each map phase, which minimizes the communication cost. The *SKY-FLT-SORT* in **[32]** uses checkpoints to filter non-skyline points and the built-in sorting mechanism of combiners/reducers to access the records/points in an ascending order based on the key value $p._x+p._y$ which is the sum of values in each dimension as defined by the *Manhattan distance which*

is also known as L_1-norm. The use of *Manhattan distance* as a distance measure has the benefit of a reduced computational cost since it avoids the additional cost of computing the *power* and especially the *square root* that exists in the *Euclidian distance (L_2-norm)*. This work uses the *Euclidian distance* and investigates if there is any performance benefit compared to the *Manhattan distance*. The access order of points with *Euclidian distance* and *Manhattan distance* is presented in **Figure 44**. To achieve sorted access on the dataset the algorithm can use any scoring function just like the distance metric. The SKY-FLT-SORT algorithm cannot fulfil all user's preferences since it cannot provide an answer out-of-the-box when the user wants for example to maximize his/her preferences. This happens because the distance measure does not take into account the preferred origin of space. This leads into points with the largest coordinates to have the largest distance from the preferred point of origin. This has also an impact on the checkpoint selection. To avoid this issue, the output key and hence the distance metric should be $D(\mathrm{p},\mathrm{o}) = \sum_{i\hat{1}D}(|p_i - \mathrm{o}_i|)$, where *o* is the origin point of preferences.

The checkpoint selection in the mapper phase of *SKY-FLT* algorithm is susceptible to the access order of records/points. In more detail, the *map* phase accesses the points as written in the indexed file. This way there are cases that the checkpoint loses its pruning ability due to continuous updates. In an extreme case, this would lead to outputting to the combiner/reducer a large number of points, or even a whole partition, even if the true output should be a single point. This extreme case could exist if the data is written in the reverse order of the user's preferences. Note that a sorted dataset that achieves the best *min-min* skyline performance would have the worst max-max skyline performance.

The pseudocode in **Algorithm** 1: describes a sort-based algorithm for skyline computation in SpatialHadoop that uses filtering points and a sorting technique. The algorithm in the first phase reads the indexed dataset from the *HDFS*. It identifies which partitions may contain candidate skyline points. Each *map* task processes the records/points of a single partition. Then, it computes the local skyline for each partition and finally, the reduce phase collects all the local skylines, merges them and outputs the result set.

The main difference from the two algorithms described in **[32]** and this proposed implementation is that the previous work shares the computation cost of the skyline in both *map* and *reduce* phases. This work, on the other hand, stresses the *map* phase a lot more in order to get local skylines which will have small cardinality **[448, 449]**. The approach of computing and merging of local skylines in *SpatialHadoop* was also followed in the *CG_HADOOP* skyline algorithm but is a usual practice in multi-core and distributed environments. This minimizes the communication cost and allows a single reducer to identify the result set from a small portion of the initial dataset. Alongside, the identification of local skylines is crucial for the reverse skyline algorithm proposed in **section 5.4.**

Algorithm 1: *SSAS*: A Sort-based Skyline Algorithm for SpatialHadoop.

```
1:   function CELLSFILTER(C: setofcells)
2:        Initialize candidatepartitionList list
3:     for all cell c∈C do
4:       if c is not dominated by candidatePartitionList then
5:           Add (c, candidatePartitionList)
6:           Update candidatePartitionList
7:         end if
8:      end for
9:    Load every c∈candidatepartitionList in a map function.
10:  end function
11: function MAP(P: Setofpoints)
12:  Initialize origin & filter points based on user's preferences and LIST.
13:   for all points p∈P do
14:     if p is notdominatedby filter then
15:         Add (p, LIST)
16:  if distance (p,origin) < distance (filter,origin) then
17:             Update filter
18:          end if
19:     end if
20:   end for
21:  SORT LIST based on the distance from origin Initialize skylist
```

```
22:   for all points p∈LIST do
23:      if p is notdominatedby filter then
24:         if p is notdominatedby skylist then
25:              Add (p, skylist)
26:              output (null, p)
27:          end if
28:       end if
29:    end for
30: end function
31: function REDUCE (null, P: Setofpoints)
32: Initialize origin based on the user's preferences and LIST.
33:  Add all p∈P   into LIST
34:  SORT LIST based on the distance from the origin
35:  Initialize skylist
36:  for all points p∈LIST do
37:      if p is notdominatedby skylist then
38:              Add (p, skylist)
39:              output (null, p)
40:       end if
41:   end for
42: end function
```

As soon as the *MapReduce job* has started, the *CellFilter (lines1-9)* reads the *global index* stored in the *HDFS*. By reading the *global index*, the algorithm (line 3) has access to the entry of all partitions that contain the data. Each of these partitions is stored as a single file in *HDFS*. At this point, a pruning process must be performed *(line 4-6)* to identify the partitions that contribute to the final answer. This identification process reads only the boundaries of each partition and not the actual data. This minimizes the read operations from the *HDFS*. Each partition is checked against the list of partitions that have been processed so far and fulfil the criteria to be a candidate, thus containing candidate skyline points. If the new partition is not dominated by any other partition in *candidatepartitionList*, it is inserted in the list removing any dominated partitions by it. In the end, every partition will invoke a *map* phase *(line 9)* that will process the data inside each partition.

Following, a *map* phase *(line 11)* will run for each partition. The order that the records/points will appear in the *map* phase is the order in which they are written in the indexed files. The *origin* point of the data space is defined by the user's preferences *(line 12)*. During the *map* phase, a *filter* point with the greatest pruning power is maintained in order to filter the dataset *(line 12)*. Initially, the *filter* point is set to be the point of the data space with the greatest distance from the *origin point (line 12)*. If the boundaries of data space are unknown, this point could be set to the maximum or minimum programmatically value. In the next step, all the points *p* reaching the map phase *(line 13)* are inserted into a *LIST (line 15)* under the condition that are not dominated by the *filter* point *(line14)*. Additionally, if the new point *p* is closer to the *origin* than the *filter* point *(line 16)*, the *filter* point is updated *(line 17)*. At this stage, the *filter* point has the greatest pruning power among all points examined since it is the closest to the *origin*. Additionally, the *LIST* has a portion of the records that have reached the *map* phase due to the pruning power from the *filter* point. By taking advantage of the *LIST's* smaller cardinality, we perform a sort operation based on the distance of points to the *origin* point *(line 21)* to ensure that the algorithm will access first the most promising point to be a *(local) skyline* point. This is the same approach followed by *BBS* algorithm to access the nodes inside its list. With a sorting operation over the dataset, we achieve to identify the candidate skyline points early in the *LIST*. A drawback of the algorithm is that it must traverse the complete sorted list. A workaround can be found in the SFS **[41]** , *LESS* **[10]**, or *SALSA* **[42]** algorithms to avoid the complete traversal of the list when the values of the dataset are in a specific domain or when the normalization of values is acceptable. Following, we check all the points in the *LIST (line 22)* if they are dominated by the *filter* point *(line 23)* and then if dominated by the *skylist (line 24)*, which contains all the *local skyline* points. At this stage, the *filter* point has the maximum pruning power leading in discarding almost every point that will not be part of this *local* skyline. If neither of the previews holds, the point *p* under examination definitely belongs to the *local skyline*, thus inserted in the *skylist (line 25)* and outputted *(line 26).* As a result, each *map* phase computes a local skyline for its assigned partition. The cardinality of the records/points of each *local skyline* will be small in comparison to the dataset and in particular $\Theta\big((\ln n)^{d-1}/(d-1)!\big)$ **[448, 449]** for uniform distribution. Furthermore, the use of a combiner will not provide any benefits since each local skyline contains the minimal set of data needed to

produce a correct answer. Since the cardinality of each local skyline is small, a *null* key will be used in order to process all the points in a single reducer. Similarly, the use of a single reducer is also the case for the *reverse skyline* computation in **section 5.4** . The *reduce* phase *(line 31-42)* collects all the *local skylines* and produces the final skyline. Since the number of points reaching the *map* phase is small, the algorithm does not maintain a filtering point. As with the *map* phase, all points are inserted into a *LIST (line 33)* and are sorted based on the distance to the *origin* point *(line 34)*. Finally, all points are checked against the list of candidate skyline points. Since the dataset to be processed is sorted, any point not dominated by the *skylist* will be a final skyline point and it is inserted into the *skylist (line 38)* to help in the pruning process and outputted *(line 39)*.

The key aspect of the algorithm is that the *CELLFILTER* performs the major pruning of the dataset. Each *map* task computes the *local skyline* of its assigned partition. The sorting mechanism guarantees that the *local skylines* computed in the *map* phase are correct. Due to the *local skylines*, the cardinality of data transferred to the *reducer* is minimal.

5.4. A Reverse Skyline Algorithm in SpatialHadoop

The previous section proposed the *SSAS skyline algorithm*. This section will build on top of *SSAS* a *reverse skyline algorithm* named *SRSAS*. As described in the preliminaries section, we can compute the *reverse skyline* on top of a two-step filtering approach with an additional refinement workload. Initially, for every *map* phase, a local *global skyline* is computed. All the *global skyline* points from each partition will form a superset of the *reverse skyline* result set. The *reduce* phase will compute the final *global skyline* of the dataset and at the end, a refinement over the *global skyline* is performed, with a second *job* to retrieve the final answer. As with the *SSAS* algorithm, the idea is to harvest the power of the *CellFilter* function provided by *SpatialHadoop* to prune every partition that is warranted not to contribute in the final solution, minimizing the amount of data to be processed.

The identification of promising partitions and points in a *reverse skyline* query is more complex than the *skyline* since the origin/query point is inside the data space and not on the boundaries as with the simple skyline algorithm. In order to present the pruning approach of the *SRSAS* algorithm, **Figure 45** illustrates a possible outcome of the partitions derived from the R-tree indexing mechanism of *SpatialHadoop*.

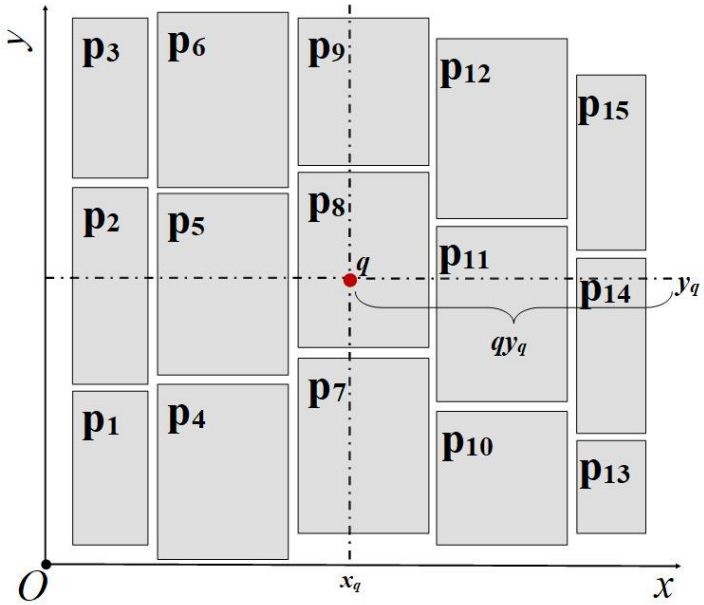


FIGURE 45: SPATIALHADOOP'S R-TREE PARTITIONING APPROACH.

In a *reverse skyline,* the query point is usually a point that does not exist in the dataset since, in the simpler scenario, the question to be answered is how the new product/service (query point) fits in the customer's preferences. Nevertheless, if the query point matches a point from the dataset, this point will be the final answer. Intuitively, the query point *q* splits the data space into

four quadrants, defined by the two parallel lines x_q, y_q to the *y-axis* and *x-axis* respectively, which intersect the query point as presented in **Figure 45**.

Based on the space division imposed by the query point *q*, a partition may be intersected by the x_q*-axis*, the y_q*-axis* or both. For the set of partitions that do not intersect with one of the x_q or y_q axes, like the partitions p_1, p_3, p_4, p_6, p_{10}, p_{12}, p_{13}, p_{14} in **Figure 45**, we can identify if they contribute to the final answer by simply performing dominance comparisons among them. For the case where a partition intersects with both x_q, y_q, like the partition p_8 in **Figure 45,** then the query point *q* will belong into this partition. This leads that the most promising points to be candidates for *reverse skyline* points and consequently *global skyline* filtering points will belong into this partition. This happens because these points will have, among all other points in the dataset, the greatest pruning power in terms of maximizing their dominance region. In the case were the partitions intersect with one of the x_q or y_q axes, like the partition p_2, p_5, p_7, p_9, p_{11}, p_{14} in **Figure 45,** there is a probability to contain candidate *reverse skyline* points near or on the axis x_q or y_q. Those points, especially the ones on top of the axis x_q or y_q, will also have among the greatest pruning power and they are guaranteed to be candidate *reverse skyline* points since they always have the best value in at least one dimension. Those points can prune partitions in both of the quadrants that belong and are the only ones to have the ability to prune partitions that intersect with the same axis and are further apart.

At this point, an important issue in the design of the *CellFilter* is that it performs the pruning based on partitions, without knowing the points contained. In a *reverse skyline* computation, this imposes restrictions to the pruning process, as previously described, in which we cannot directly prune the partitions that are intersected by the x_q*-axis* or y_q*-axis* since it is not possible to guarantee that they do not contribute to the final solution. This also occurred in the original BBRS algorithm of **[14]**.

As an example, consider the partitions p_8, p_{11} and p_{14} in **Figure 45** that are intersected by the y_q-axis as defined by query point *q* and infinity (qy_q). Intuitively, the further a point is from the query/origin point *q,* the closer to the axis it must be in order to be a *skyline* or a *reverse skyline* point. If a point exists on top of the y_q-axis in p_{11}, as shown in **Figure 46**, it could prune partitions p_{13}, p_{14}, p_{15} and all the points on the right of the dashed line that intersects it, but we are not able to identify it within the *CellFilter* without performing further processing in the partition.

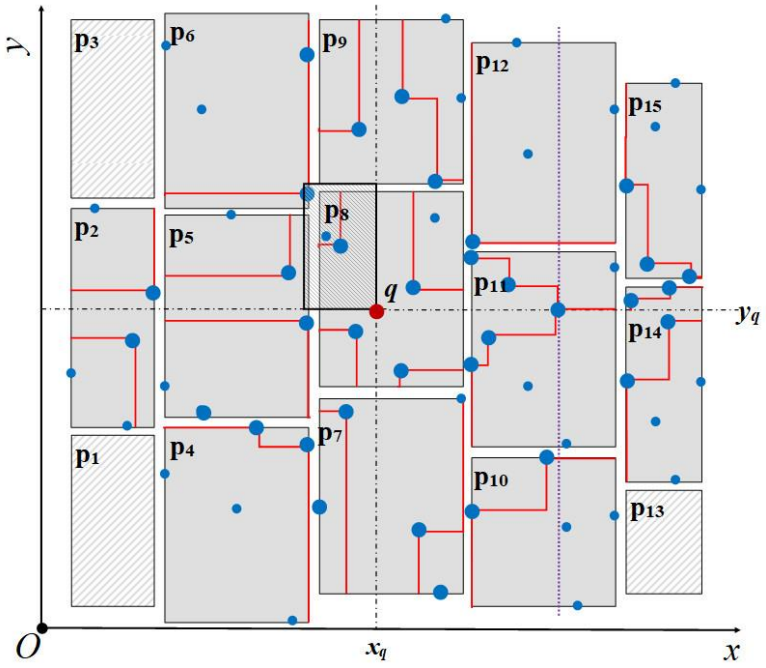


FIGURE 46: LOCAL GLOBAL SKYLINES IN SRSAS ALGORITHM.

Up to this point, the partitions intersected with the axis have not helped in the pruning process because we are unaware of the location of points inside them. Additionally, the MBRs in *SpatialHadoop* follows a different approach from the one in R-trees since there is no guarantee

that a point exists in the lower left or in the upper right corner of the MBR. By the design of *SpatialHadoop,* it is guaranteed that at least one point exists on each edge of the MBR. Based on the previous facts, the *CellFilter* of *SRSAS* assigns to *map* tasks all the partitions that are intersected by an axis and from the rest of the partitions the ones that are not *globally dominated* with each other. To compute whether or not a partition *globally dominates* another partition, we follow a similar approach to the one used in the *CellFilter* of *SSAS, SKY-FLT* and the r-tree based *BBS* algorithm. That is, we represent each of the two partitions with their vertex (point) that is closer to the query point. Using those points, one for each partition, the query point and **Definition 8** we can identify if a partition *globally dominates* another one**.** More precisely, based on **Figure 46** the *CellFilter* initially receives all the partitions. At first, it outputs the partitions p_2, p_5, p_7, p_8, p_9, p_{11} and p_{14} that are intersected by an axis. Next, for the rest of the partitions p_1, p_3, p_4, p_6, p_{10}, p_{12}, p_{13} and p_{15} it must check and discard every partition that is *globally dominated*, with respect of the query point q and output the rest. Thus, partitions p_1, p_3 *and* p_{13} are discarded since they are *globally dominated* by partitions p_4, p_6 and p_{10} respectively allowing partitions p_4, p_6, p_{10}, p_{12} and p_{15} to be outputted, and assigned to a *map* task for further process, along with the partitions intersected by an axis. Further pruning mechanisms could be designed if we were aware of the location of points inside the partition, but this would require additional cost for query processing and multiple *MapReduce jobs*.

The rest of the *SRSAS* algorithm follows the same general approach as *with the SSAS.* Just like *SSAS,* after the pruning imposed by the *CellFilter,* the mapper handles the remaining partitions. In the case of the *reverse skyline,* the mapper is dedicated to compute the local *global skylines* (**Definition 9:**) for each partition. When the *map task* assigned to this partition finishes, all the data outputted would belong to the *global skyline* of that partition. Since the *global skyline* is a superset **[14]** of the *reverse skyline*, its computation will make a great refinement on the dataset. Following the same approach as with the *SSAS* in order to identify all the *global skyline* points of a partition in one *map task,* we must sort the points based on their distance from the query/origin point. Note that the query/origin point may be outside of the partition under examination and thus the origin point of space, to which the *global skyline* will be computed, will not match with any of the edges of the partition. Nevertheless, this does not create any implication. Intuitively, for the partitions that do not intersect with the axis x_q or y_q, the result of a *global skyline* will be seen as a simple *skyline*, since all the points of that partition will be contained in one quadrant according to the axis derived by the query point q. With the same logic, if the partitions are intersected by one of the axis x_q or y_q, the *global skyline* will appear as two disjoint *skylines,* while if the query point resides inside a partition, the *global skyline* on this partition will appear as four disjoint *skylines*, as presented in **Figure 46**. As with the *SSAS* algorithm, by computing the local *global skyline* inside each map phase, the total output records to be transferred in the reducer will be minimized.

Following, the reducer will collect the output records from all the mappers. The set of collected data will be a superset of the *reverse skyline* since all points belong to the *global skyline* of the same partition. In order to compute the correct final *global skyline* in one iteration, we must sort the data and perform a global dominance comparison on each point. To retrieve the *reverse skyline*, a final refinement process must be performed in the form of a range queries as in the original algorithm **[14]**. One restriction is that in order to retrieve the minimal superset of the *reverse skyline query* in the form of a *global skyline query,* the algorithm needs to know all the *reverse skyline* candidates from all the partitions that were processed. Thus, the *reduce* phase cannot be performed in more than one task.

The pseudocode in **Algorithm** 2: and **Algorithm** 3: describes the sort-based algorithm for *reverse skyline* computation in *SpatialHadoop*, which incorporates the power of *global skyline*, a sorting technique and a refinement process in the form of range queries.

Algorithm 2: *SRSAS*: A Sort-based Reverse Skyline Algorithm for SpatialHadoop.

```
1: function CELLSFILTER(C: setofcells)
2:   Initialize requisiteypartitions list
3:   Initialize candidatepartitions list
4:   Compute xq and yq axis
5:     for all cell c∈C do
6:       if c is intersected by xq and yq then
5:          Add (c, requisiteypartitions list)
```

```
7:       else if c is not globalydominated by candidatePartitions then
8:           Add (c, candidatePartitions)
9:           Update candidatePartitions
10:      end if
11:     end for
12:  Load every c∈candidatepartitions and c∈requisiteypartitions in a map function.
13: end function
14: function MAP(P: Setofpoints)
15:  Initialize four filter points f_ll, f_lr, f_ul, f_ur, one for each quadrant, one temporary
      filter point, and a MAPLIST.
16:  for all points p∈P do
17:   Identify the quadrant of p and assign to filter the corresponding value of
       f_ll, f_lr, f_ul or f_ur
18:                 if p is not globalydominated by filter then
19:         Add (p, MAPLIST)
20:        if distance (p,query) < distance (filter,query) then
21:           Update the corresponding value of f_ll,  f_lr, f_ul or f_ur
22:        end if
23:     end if
24:   end for
25:      SORT  MAPLIST  based  on  the  distance  from  query  point  and  initialize
        localglobalskylist
26:   for all points p∈MAPLIST do
27:    Identify the quadrant of p and assign to filter the corresponding value of
       f_ll, f_lr, f_ul or f_ur
28:     if p is not globalydominatedby filter then
29:        if p is not globalydominated by Localglobalskylist then
30:          Add (p, localglobalskylist )
31:          output (null, p)
32:        end if
33:     end if
34:   end for
35: end function
36: function REDUCE (null, P: Setofpoints)
37:    Initialize REVLIST & candidaterevskylist
38:    Add all p∈P   into REVLIST
39:    SORT REVLIST based on the distance from the query point
40:    for all points p∈REVLIST do
41:       if p is not globalydominated by candidaterevskylist then
42:                 Add (p, candidaterevskylist )
43:              output (null, p)
44:      end if
45:    end for
46: end function
```

Since the *SRSAS* algorithm follows a similar approach to the one of *SSAS*, we will highlight only the key differences. The algorithm in the CellFilter (*line 1-13*) has access to the *global index*, which is the entry of all partitions that contain the data. In addition to the *candidatepartitions* list *(line 3)* found in *SSAS,* the *SRSAS* algorithm maintains a *requisiteypartitions* list in which the partitions intersected by the x_q and y_q axis are stored. The axis x_q and y_q axis are computed as a rectangle with zero height (*line 4*). Following, the algorithm iterates through all the partitions (*line 5*). If a partition is intersected by the x_q or y_q axis, it is placed in the *requisiteypartitions* list (*line 5-6*), otherwise it is checked if it is *globally dominated* (*line 7-10*), just like with simple dominance in the *SSAS* algorithm. After iterating all partitions of the *global index*, the algorithm assigns each one of the partitions in *candidatepartition* and *requisiteypartitions* list to a *map* task (*line 12*).

The *map* phase in *SRSAS* maintains four filtering points (*line 15*), one for each quadrant defined by the x_q and y_q axis instead of one in *SSAS.* For every point *(line 16-24)* in a *map* task, it is identified in which quadrant it belongs *(line 17)*. The global dominance is performed only between the retrieved point and the *filter* point that corresponds to the point's quadrant. If needed only this point is updated *(line 20-22)*. When the iteration over all points finishes each *filter* point will have the greatest pruning power in its quadrant. A sorting operation in the *MAPLIST (line 25)*, based on the distance to the query point, will guarantee that the most promising points will be accessed first. A second iteration (*line 26-34*) will produce the local *global skyline*. In this step, most of the point will be pruned from the *filter* point (*line 28*) minimizing the need to iterate through the candidate points of *localglobalskylist (line 29).*

The reduce phase (*line 36-49)* collects the points from all *map* tasks into a *REVLIST (line 38)* and sorts them based on the distance from the query point *(line 39)*. With an iteration over all the points in the *REVLIST (line 40-45),* the final *global skyline* of the dataset is produced and written to the *HDFS.*

As with the *BBRS* algorithm, the final check that every point in the *global skyline* should pass in order to be true *reverse skyline* points involves a *range query* [14]. The MBR of each *range query* has its center over the *global skyline* point under examination and its edge on the *query point* of the *reverse skyline,* as shown in **Figure 46**. The query is performed in order to identify if there is any point in the window defined. If no point is found, the *global skyline* point under examination is promoted to a *reverse skyline* point. In our approach, we make use of the built-in *range query* algorithm of *SpatialHadoop* in a final refinement process.

Algorithm 3: Refinement process of the *SRSAS* algorithm.

```
1: function RANGEREFINEMENT(GS: globalskylineset)
2:     Initialize revskylist
3:     for all points p∈GS do
4:             Compute the range area r_pi based on p and query point
5:        execute the range query(r_pi)
6:          if range query is empty and p is not globally dominated by revskylist then
7:              add p to the revskylist;
8:                end if
9:      end for
10:     output (revskylist);
11: end function
```

The refinement process (**Algorithm** 3:), for every point that belongs to the *global skyline* set *GS* (line 3), accessed in sorted order of their distance from the origin, we compute the *range area* (line 4) that will be used in the *window query*. Each one of the *range areas*, in the form of a rectangle parallel to the axes, is computed using a global skyline point and the query point as defined in [14]. The *global skyline* point must be in the center of the rectangle (the point where the diagonals intersect) and the query point must be a vertex of this rectangle. Given those two conditions, a single rectangle can be computed which will represent the *range area*. Then, the algorithm performs a *window query* for the given *range area* over the initial dataset (line 5). To accomplish this task, the refinement process of *SRSAS* uses the *range query* that is integrated into *SpatialHadoop,* which spawns one new job for each range query. The *range query* uses only the *map* function since the existence of a point can be identified with the use of the built-in map counters of *MapReduce*.

As with the *SSAS* algorithm, the key aspects of the *SRSAS* algorithm is that the *CELLFILTER* performs the major pruning of the dataset. Each *map task* computes the *local global skyline*, which assists in minimizing the data needed to be processed by the *reducer.* The sorting mechanism in the *map* and *reduce* phase, along with the range refinement process guarantees that the final answer is correct.

5.5. Experiments

This section studies the performance and effectiveness of our proposed algorithms for *skyline* and *reverse skyline* computation. For the evaluation, we used synthetic and real datasets over a distributed and pseudo-distributed platform. The performance metrics involve the *time cost*, in terms of the total time spent to retrieve an answer and the time spent in the various phases of our algorithms along with the amount of *data transfers* needed. All algorithms were written in *Java* and the *source code*[10]along with the *commands* to recreate the datasets are available under the *Apache License 2.0.* For the accuracy of our results, we ran each experiment 10 times and the average cost of all tries was estimated. We compared our *SSAS* skyline algorithm with the SKY-FLT algorithm from [32], which performed better in comparison to the SKY-FLT-SORT and the CG-HADOOP in large datasets. In the rest of the section, it is shown that our proposed *SSAS*

[10] https://github.com/ChristosKalyvas/SkylineQueriesInSpatialHadoop

algorithm improves over the SKY-FLT algorithm with up-to 15%. The case of the *reverse skyline* in *SpatialHadoop* is also studied for the first time.

The deployment, management and administration of Hadoop clusters was performed with the open source Apache Ambari 2.5 management platform over CentOS 6.4 Linux nodes. The communication of nodes was performed over a Gigabyte network. The version of SpatialHadoop was 2.4 and the installation involved adding the dedicated libraries to the *Hadoop* installation directory of each node. The infrastructure consisted of 4 physical machines, which hosted 6 virtual nodes. Each node had 16 logical processors with 16GB of RAM. We additionally compared our algorithms over a pseudo-distributed environment in order to isolate any factor that may affect the accuracy of the results. The pseudo-distributed node consisted of 32 logical processors with 32GB of RAM.

The datasets that were used are three synthetic and one real dataset. The official *SpatialHadoop* Site[11] provides numerous *real* datasets including many datasets extracted from *OpenStreetMap* **[450]**. A study that presents a large number of *real* datasets can be found in **[451]**. The *real* dataset used in this work is the *All nodes*[12] dataset of OpenStreetMap **[450]** which holds all the points on the planet and contains 2.7 Billion records with an initial size of 96 GB. The initial dataset was refined to hold only the coordinates of each point that lead to a size reduction to 60 GB. This dataset is the newer version of the one used in **[32]**. The synthetic datasets were created with the random spatial data generator of *SpatialHadoop*. The distribution of each synthetic dataset was *uniform*, *correlated* and *anti-correlated* respectively with 2.7 Billion points each and a size of approximately 81 - 134 GB. For the *correlated* datasets, the selected ρ-value (rho-value) was 0.8 in order to produce a dataset that will give a large number of skyline points. The ρ-value defines the correlation between the variables of our dataset, which in term defines the way the points generated over the data space.

To investigate the scalability of our algorithms, for each of the four datasets with 2.7 Billion records we produced 4 smaller datasets of 1M, 10M, 100M and 1B records with the process of sampling. The size of those datasets varied from 25 MB to 134 GB. The smaller 1M dataset was used to investigate how the algorithms performed in local mode without using the *MapReduce* framework. The sampling process was executed with the built-in method of *SpatialHadoop*. All datasets were indexed by an R-tree. The number of partitions in each index varied from 1 to 900 depending on the dataset. For the indexing process, we used the default values of the parameters. That includes the value of *spatialHadoop.storage.IndexingOverhead*, which controls the number and size of partitions to be 0.1 and the value of *spatialHadoop.storage.SampleRatio*, which controls the quality of the index in terms of time cost and memory to be 0.01.

Since a sorting operation on a large dataset may not be desirable or even feasible, *SpatialHadoop* has the ability to index the dataset in smaller partitions. This approach is useful in low memory systems since every *map* phase will have to sort a smaller portion of the dataset. The use of smaller partitions would also be beneficial to the *CellFilter* function since the number of partitions would increase but each partition would have fewer data. The increased number of partitions would lead to a better partition-level pruning of the dataset. A partition related parameter of *SpatialHadoop* that can be used for this purpose is the *spatialHadoop.storage.IndexingOverhead.* By default, it is set to 0.1 meaning that for 1GB of dataset, 100mb additional data, related with partition boundaries, will be written. For an even more efficient partitioning, the selection of a *spatialHadoop.storage.SampleSize* parameter can result in a more accurate representation of the dataset before the initialization of the indexing mechanism. In order to identify our datasets we used the number of records instead of the dataset size as a descriptor, since the file size may not always be an appropriate measure. For example, consider two datasets with 1M records each, where the first stores the points with a precision of two decimal points and the other with six. The file size of the second one will be larger but the records will be the same.

[11] http://spatialhadoop.cs.umn.edu/index.html

[12] http://spatialhadoop.cs.umn.edu/datasets.html

5.5.1. The case of the SSAS algorithm

In order to compare the performance of our algorithm we implemented and used the SKY-FLT algorithm from [32]. In their work, the authors tested their algorithms in minimizing all preferences (min-min mode) in an environment with strictly non-negative values and through their research they state that the SKY-FLT performs better in the largest of their datasets. In this work, we equipped the SKY-FLT algorithm with the *Euclidian distance* metric, which makes it applicable in the rest of the modes and in negative and non-negative value environments without any implication. The SKY-FLT-SORT, on the other hand, needs more attention in environments with negative values like the *real* dataset used in this work due to its construction. Moreover, the author's research belongs to the 1M-100M range of our datasets where the 1M case is our implementation activates the pure local approach without the use of the *MapReduce* environment. With this in mind, we consider that SKY-FLT is the appropriate algorithm to be compared with ours.

Hadoop, by design, reads a vast amount of data from HDFS, even when reading a single block. As an example, each one of the approximately 900 blocks in our 2.7B records datasets holds on average 3M points. Thus, the order that the records are written in the files has an impact on the time needed to retrieve an answer. In *SpatialHadoop's* case, the points inside a block of the R-tree are written in a partially ascending sorted order of their first and second coordinate value. The ordering in which the partitions, produced by *SpatialHadoop,* as presented in **Figure 45** and written to the master index, also depicts this ordering. This ordering has an impact on the computation time of the skyline under various preferences, as presented in **Figure 47.** Through our experimentation, the use of *Euclidian distance* had no significant performance gain in comparison to the *Manhattan distance*.

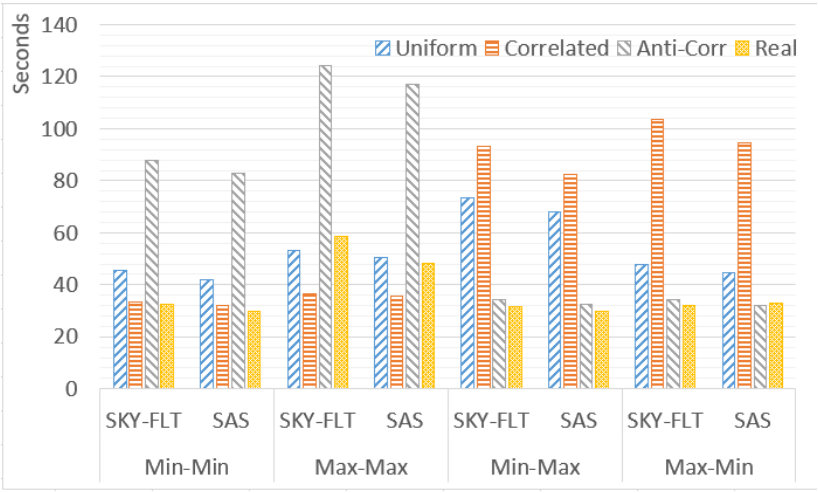


FIGURE 47: EXECUTION TIME OF SKY-FLT AND SAS OVER THE 100M DATASETS.

More specifically, in a 2-dimensional environment there can exist four preferred modes, in terms of minimizing or maximizing the values on each dimension, named *min-min*, *max-max*, *min-max*, *max-min*. For the case of the *uniform* dataset, the *min-max* case needs approximately 30% more time than the case of *min-min* in both algorithms. The same holds in computing the equivalent min-min case and the *max-max* case in the *anti-correlated* dataset.

The difference of the *min-min* mode in the *anti-correlated* dataset in comparison to the equivalent min-max mode of the *correlated* dataset is more clear in the 2.7B dataset presented in **Figure 48** in which there is a 7% difference in time. At this point, we show that the mode of operation has an impact on the computation time of both algorithms due to the order in which the points are stored in the index file. This could potentially affect the computation of reverse skyline and convex hull since certain mappers may need more time to accomplish their task delaying the whole process. Furthermore, note that the case of the *real* dataset seems to perform equivalently in both 100M (**Figure 47**) and 2.7B (**Figure 48**) datasets, which will be discussed further in this section.

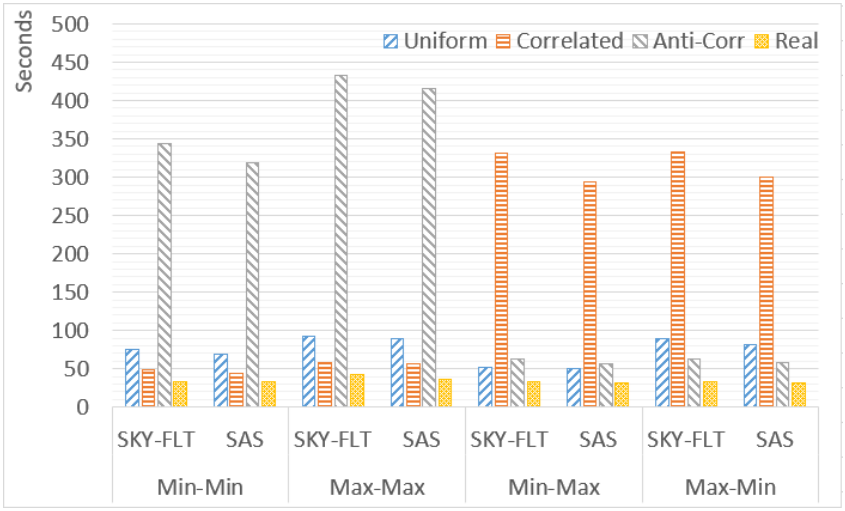


FIGURE 48: EXECUTION TIME OF SKY-FLT AND SAS OVER THE 2.7B DATASETS.

At this point, we will discuss the time cost of computing the skyline with both SKY-FLT and SAS algorithm over all four datasets in our pseudo-distributed environment. The evaluation was performed over the complete datasets as well as in the five sampled versions of each original dataset. In addition to the *min-min* mode, we present the cost to compute the skyline over all four modes.

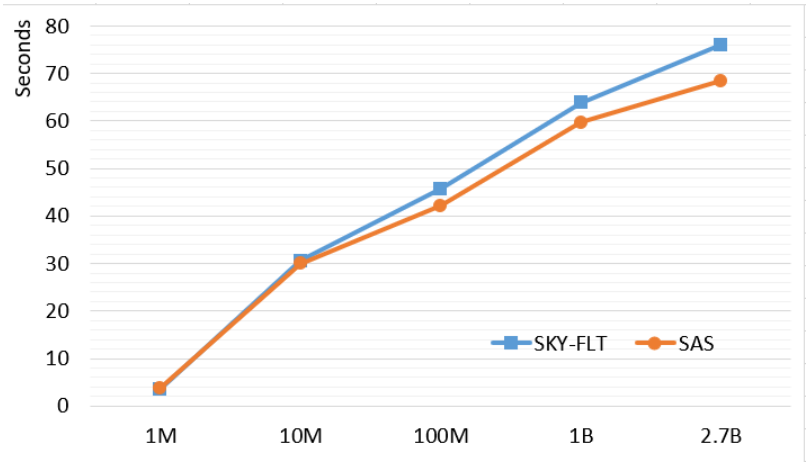


FIGURE 49: EXECUTION TIME OF SKY-FLT AND SAS IN MIN-MIN MODE OVER THE UNIFORM DATASETS.

In **Figure 49** the time needed by both SKY-FLT and SAS algorithms to compute the skyline in *min-min* mode is presented. The SAS algorithm outperforms the SKY-FLT in the datasets over 10M. In the 2.7B case, SAS performs 10% better than the SKY-FLT. The same improvement appears in the *max-min* case. For datasets smaller than 10M, SKY-FLT is slightly faster.

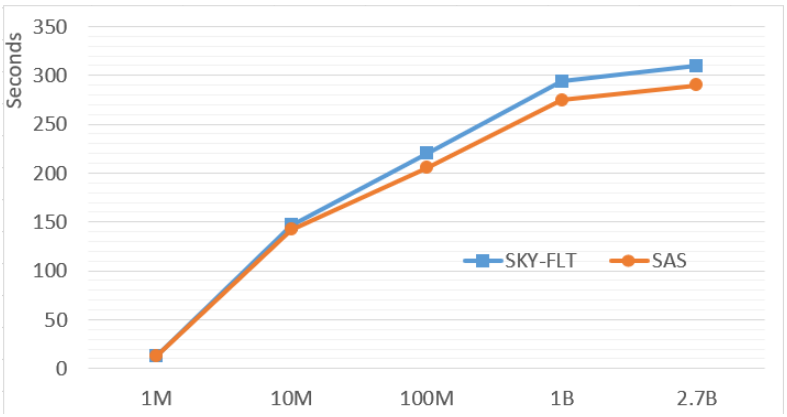


FIGURE 50: EXECUTION TIME OF SKY-FLT AND SAS IN ALL MODES OVER THE UNIFORM DATASETS.

The *total time cost* as the sum of the *time cost* to compute *all four modes* of the skyline is presented in **Figure 50.** As previously, SAS outperforms SKY-FLT in datasets over 10M points. The improvement of SAS on *all modes* over SKY-FLT, in the 2.7B dataset, is 7% percent. Both SKY-FLT and SAS seems to reach a *time cost* limit as the dataset size increases. This is due to the *CellFilter* function that is part of SpatialHadoop.

An initial implementation of *CellFilter*, related to the skyline queries, can be found in the CG-HADOOP algorithm. Its BNL-like implementation works quite efficiently since it needs only 3 milliseconds to select the partitions even in the 2.7B datasets which has 900 partitions. An alternative implementation could be useful only in datasets with hundreds of thousands or even million partitions. This can occur in extremely large datasets or in datasets where the indexing factors are set to produce a large number of partitions.

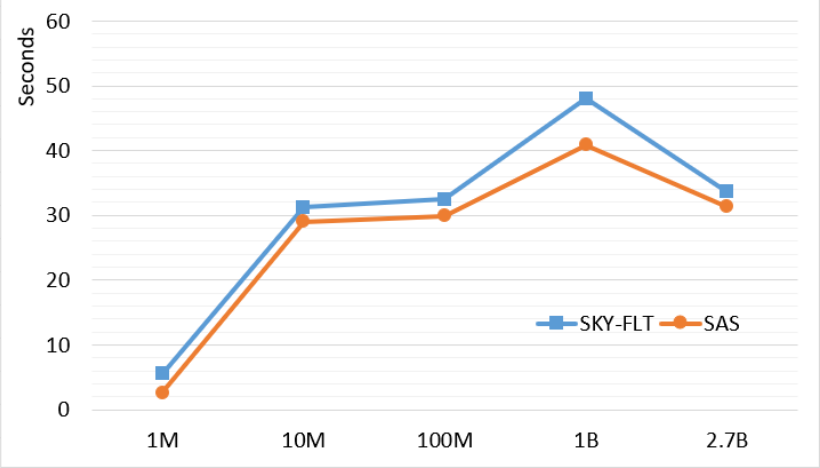


FIGURE 51: EXECUTION TIME OF SKY-FLT AND SAS IN MIN-MIN MODE OVER THE REAL DATASETS.

The *time cost* for computing the skyline in the *real* dataset is presented in **Figure 51**. The improvement of SAS over the SKY-FLT in the *min-min* case of the 1B dataset is 14% while in the *max-max* case is 15%. The *time cost* in the *real* dataset has many fluctuations. More particularly, the *time cost* in 10M and 100M, for both algorithms, is approximately the same. In both cases, the *CellFilter* function returns a single partition. The difference is that in the 10M dataset each partition holds 3.5M points and the 100M dataset 4M points. The *time cost* in the case of the 2.7B dataset is also similar to this of the 10M and 100M datasets because, as previously only one partition is processed, but in this case with 4.8M points. For the case of the 1B dataset, four partitions are returned with 5M points each.

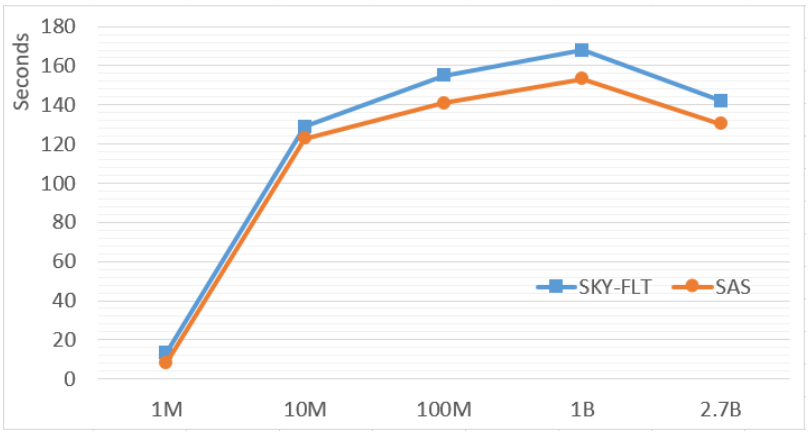


FIGURE 52: EXECUTION TIME OF SKY-FLT AND SAS IN ALL MODES OVER THE REAL DATASETS.

The *overall cost* in computing the skyline in *all modes* over the *real* dataset, as presented in **Figure 52**, follows a *time cost* similar to the *uniform* case with the exception of the 2.7B dataset. In the 2.7B case, a single partition dominates all the other ones and this happens in every mode. This behaviour appears due to the type of dataset and the existence of points that could be considered as outliers. The improvement of SAS over SKY-FLT in the 2.7B dataset is 7% in *min-min* mode, 12% in *max-max* mode and 8% on average over *all modes*.

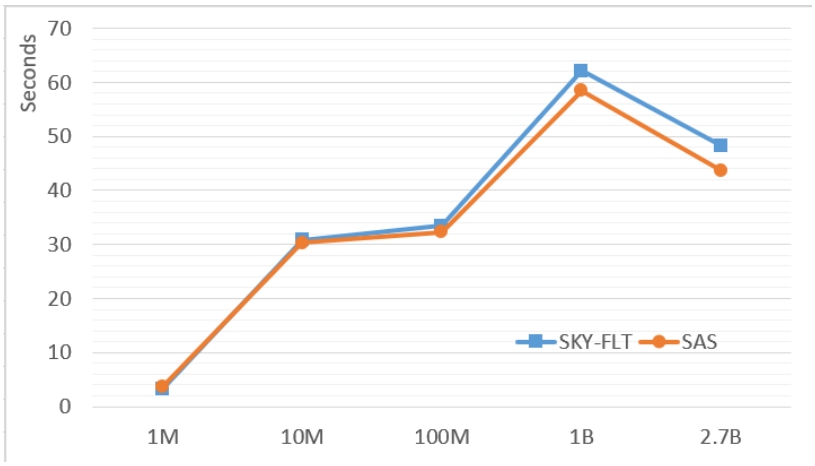


FIGURE 53: EXECUTION TIME OF SKY-FLT AND SAS IN MIN-MIN MODE OVER THE CORRELATED DATASETS.

For the case of the *correlated* dataset, as presented in **Figure 53,** the 10M and 100M cases handle 1.7M and 3.8M points respectively, which belong to more than one partitions. Interestingly, both algorithms perform better in the larger 2.7B dataset rather in the smaller 1B dataset. This happens because the *CellFilter* function outputs in total 15M points for the case of the 1B dataset and 12M points for the case of the 2.7B dataset. The improvement of SAS over SKY-FLT in the case of the 2.7B dataset is 9%.

At this point, we omit presenting the *total cost* in *all modes* of the *correlated* and *anti-correlated* dataset since computing the *min-max* and *max-min* case in the *correlated* dataset is computationally equivalent to compute the *min-min* and *max-max* case on the anti-*correlated* dataset, due to their specific distributions. The same holds with the *min-max* and *max-min* case of the *anti-correlated* dataset which is computationally equivalent to the *min-min* and *max-max* case of the *correlated* dataset. For that reason following we will present the last remaining case of computing the *min-min skyline* over the *anti-correlated* dataset.

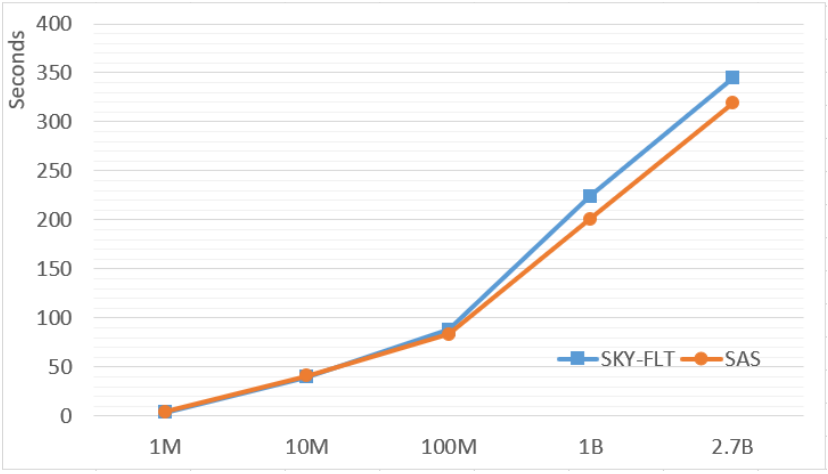


FIGURE 54: EXECUTION TIME OF SKY-FLT AND SAS IN MIN-MIN MODE OVER THE ANTI-CORRELATED DATASETS.

The computation of *min-min* and *max-max* skyline in an *anti-correlated* dataset is one of the most expensive operations. This happens because the dominance comparisons are not capable of pruning the dataset enough even with the existence of the *CellFilter* function. More particularly, the map phase for the 1B and 2.7B datasets needs to handle 27 and 44 partitions respectively, which is up to 130M points. As presented in **Figure 54**, as the dataset size increases, the number of points needed to be processed grows rapidly which affects the *time cost*, especially for the cases of 1B and 2.7B datasets due to the large number of spilled records. The overall gain of SAS over SKY-FLT in this scenario is 7.5% in the 2.7B dataset. From the previous discussion, we conclude that the BNL-like lists that maintain candidate skyline points in SKY-FLT incur a considerably larger overhead than a sorting mechanism in SAS.

As previously discussed, a key part of both *SKY-FLT* and *SAS* algorithms is the *CellFilter* function, which is the first pruning mechanism. The *CellFilter* function performs the same in both the pseudo-distributed and distributed environments. Following, we present, for the case of *min-min* mode, the percent of dataset that is outputted by the *CellFilter* and that will be processed by the numerous *map tasks*.

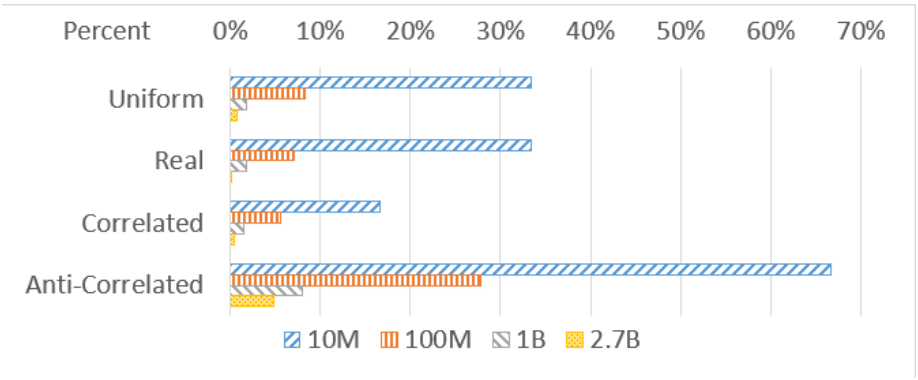


FIGURE 55: TOTAL NUMBER OF POINTS IN THE OUTPUT OF CELLFILTER AS A PERCENT OF THE INITIAL DATASET IN MIN-MIN MODE.

The *CellFilter* function performs better as the number of partition increases. As presented in **Figure 55** for the 10M dataset, 33% on average or 2 out of 6 partitions must be handled by the *map* phase. As the size of the dataset grows along with the number of partitions, the percent of the dataset to be processed by the *map* phase declines. The hardest case among all is the one of the *anti-correlated* dataset where the dominance comparisons between partitions do not prune many partitions due to the specific distribution of the dataset. In the case of the 2.7B dataset less than 5% of the dataset is handled to the *map* phase, less than 8% in the case of the 1B dataset and less than 8.5% for the 100M dataset with the *anti-correlated* case as an exception. An improvement, especially for on the smaller datasets, can be achieved by setting the appropriate parameter of the indexing mechanism to produce smaller partitions in size, with the drawback of additional storage cost.

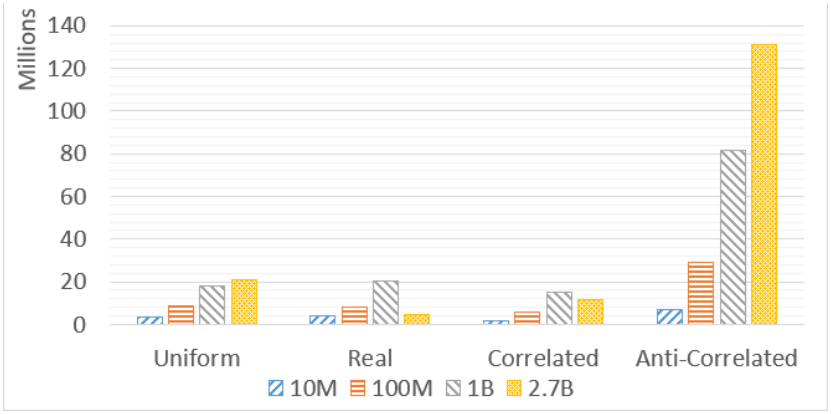


FIGURE 56: TOTAL NUMBER OF POINTS IN THE OUTPUT OF CELLFILTER.

The total number of points to be processed by the *map* phase is reverse analogues to the percent of the dataset needed to be processed, according to **Figure 56**. Even if the whole 10M dataset needs to be processed is in fact quite small in comparison to a portion of the dataset that is processed in one of the larger datasets. In the worst case of our experiments, for the 2.7B *anti-correlated* dataset, a total of 130M points are handed over to the *map* phase. In all other cases, the algorithms need to handle at most 20M points with the case of the 1B anti-correlated dataset to be at 81M. An indexing approach with smaller partitions, with the help of dominance comparisons in the *CellFilter* function, would drastically decrease those numbers.

In the following section, we compare the *time cost* of computing the min-min *skyline* in the pseudo-distributed environment over the distributed environment with the SAS algorithm. We omit the case of the 1M dataset since, as previously mentioned, this dataset was selected to explore only the local aspects of the algorithms.

The time cost to compute the *min-min* skyline over the *uniform* dataset in the pseudo-distributed environment increases, as the dataset gets larger following a linear approach. The computation of *min-min* skyline in the distributed environment remains almost constant, as presented in **Figure 57**. For the distributed case, this is also an indication that the number of partitions produced by the *CellFilter* are evenly handled to the nodes in the distributed environment.

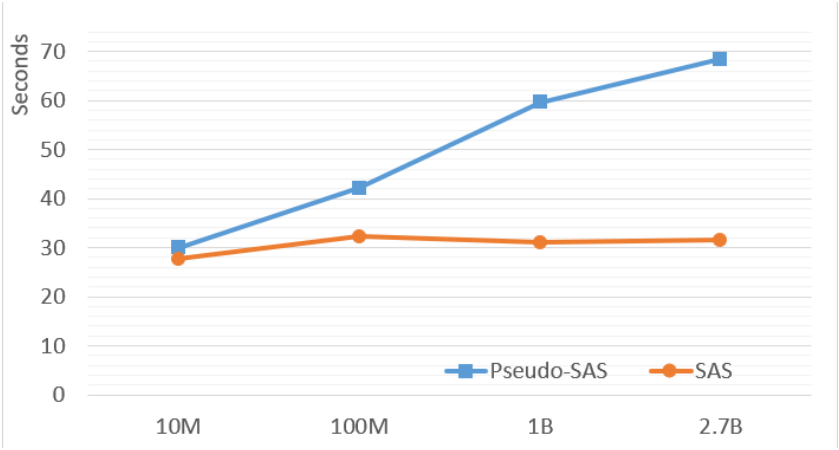


FIGURE 57: EXECUTION TIME OF SAS TO COMPUTE THE SKYLINE IN DISTRIBUTED AND PSEUDO-DISTRIBUTED MODE OVER UNIFORM DATASET IN MIN-MIN MODE.

The comparison of the time cost to compute *the min-min* skyline over the *real* dataset in the pseudo-distributed and distributed environment, as presented in **Figure 58**, reveals that the data size received from the *CellFilter* is small enough to allow the pseudo-distributed node to reach the efficiency of the fully distributed cluster. An exception to this is the 1B case where the pseudo-distributed node handles a large number of points and thus produces a large number of spilled records. Those records diminish in the 2.7B dataset due to a new skyline point that exists in the

larger dataset which prunes effectively the aforementioned points in the *CellFilter* function or in the *map* phase.

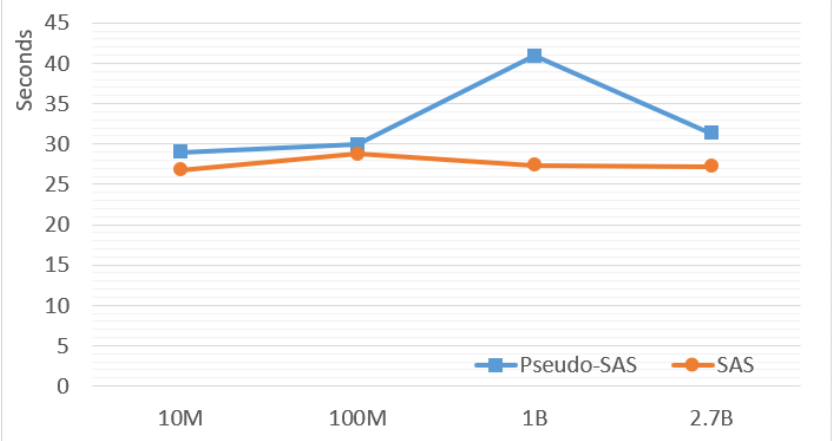


FIGURE 58: EXECUTION TIME OF SAS TO COMPUTE THE SKYLINE IN DISTRIBUTED AND PSEUDO-DISTRIBUTED MODE OVER THE REAL DATASET IN MIN-MIN MODE.

In comparison to **Figure 51,** we conclude that the SKY-FLT algorithm, with the 1B dataset, produces more spilled records than the SAS. This happens in the cases where the pruning power of the filtering mechanisms in the *map* phase of SKY-FLT degrades leading to continuous checkpoint updates due to the tuple ordering in HDFS files. In this case, the *map* phase outputs a large amount of data that triggers the combiner. In the SAS algorithm, there is no need for a combiner since the cardinality of the local skyline from each *map* phase is small.

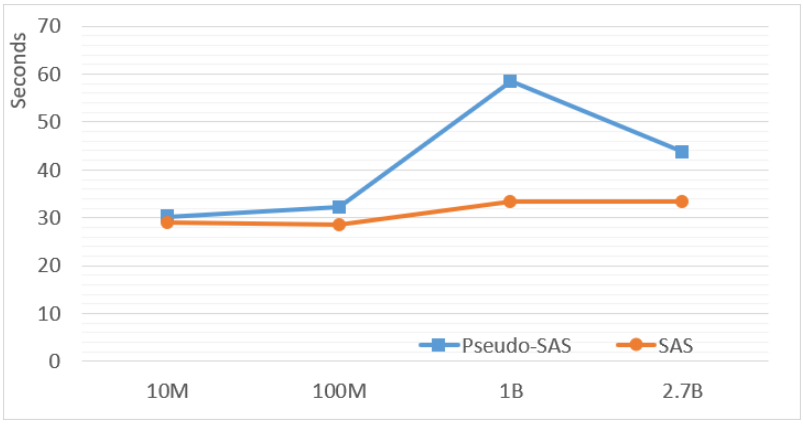


FIGURE 59: EXECUTION TIME OF SAS TO COMPUTE THE SKYLINE IN DISTRIBUTED AND PSEUDO-DISTRIBUTED MODE OVER THE CORRELATED DATASET IN MIN-MIN MODE.

A similar time cost, with the *real* dataset presented previously, can be seen in the c*orrelated* dataset as presented in **Figure 59**, which is expected since in this type of dataset the *CellFilter* works efficiently producing the least amount of data. As previously stated, the case of the 1B dataset produces more points than the case of the 2.7B dataset and this leads to more spilled records. In all other cases, the size of the data output is small enough and comparable to the one produced by the *real* dataset. Nevertheless, the *correlated* case seems to handle more points than the *real* dataset and that is confirmed from the slightly increased time cost of the c*orrelated* dataset in comparison to the equivalent case of the *real* dataset.

When conducting our research we did not expect that the *real* dataset would have a similar and even smaller time cost to the c*orrelated* dataset. Through our experimentation, we conclude that this behaviour is due to the large number of points that exist in the *real* dataset and act as outlier points. Those points, especially in the *min-min* case, prune the majority of the dataset due to their positions inside the dataset. Nevertheless, this behaviour does not exist in the reverse skyline queries and may be absolute in the case of constrained skyline queries where the computation of skyline is performed in a restricted area of the dataset.

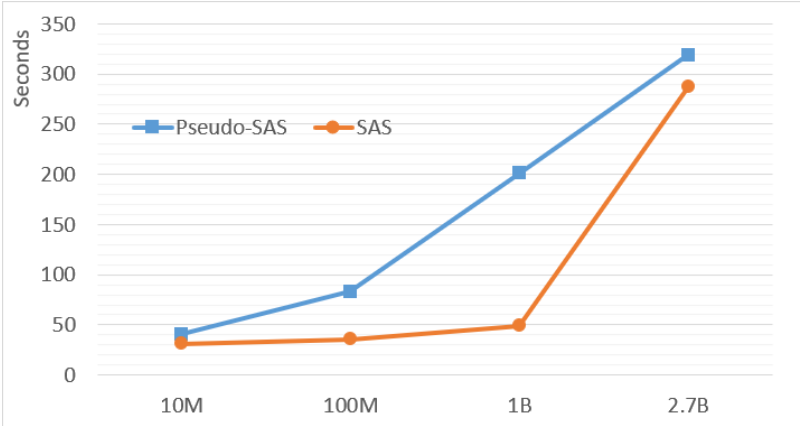


FIGURE 60: EXECUTION TIME OF SAS TO COMPUTE THE SKYLINE IN DISTRIBUTED AND PSEUDO-DISTRIBUTED MODE OVER AN ANTI-CORRELATED DATASET IN MIN-MIN MODE.

The computation of the *min-min skyline* over the *anti-correlated* dataset is presented in **Figure 60**. The distributed cluster seems to handle quite well the large number of data needed to be processed in this type of dataset. An exception is the case of the 2.7B dataset for which the capabilities of our cluster are exceeded, leading to abnormal behaviour. A solution to improve the time cost, in this case, is to add more nodes or configure the indexing mechanism to produce more, smaller in size, partitions which would lead to a smaller set of data to be processed by each map task.

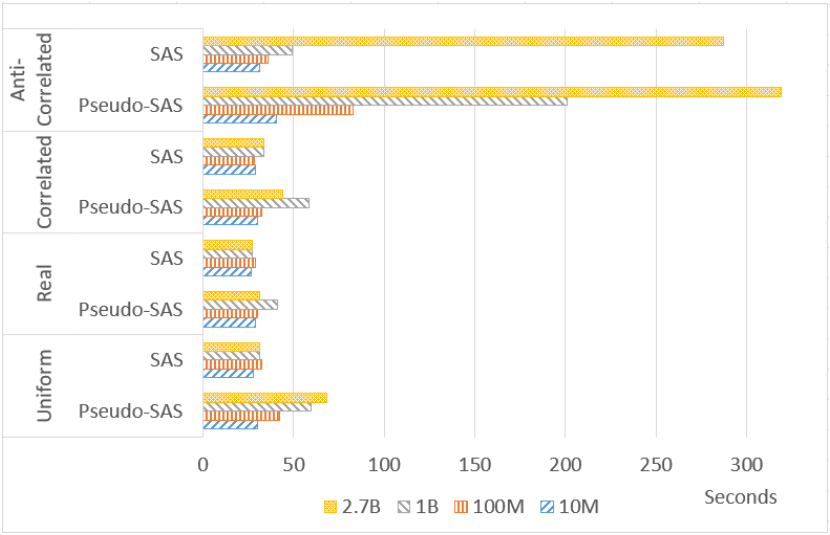


FIGURE 61: EXECUTION TIME OF SAS TO COMPUTE THE SKYLINE IN DISTRIBUTED AND PSEUDO-DISTRIBUTED MODE OVER ALL DATASETS IN MIN-MIN MODE.

A summary on the time cost of the SAS algorithm in a pseudo-distributed and distributed environment over all datasets and their sampled versions is presented in **Figure 61**. The true benefit of a real distributed cluster is visible in the *anti-correlated* and *uniform* dataset. In the majority of cases, the time cost in a distributed environment is between 20 and 50 seconds except for the single case of the anti-correlated, 2.7B dataset. In the pseudo-distributed environment, for the majority of cases, the time cost is between 20 and 75 seconds without considering the *anti-correlated* dataset for which the time cost starts from 88 seconds in the 100M version of the dataset.

A further reduction in the time *cost* of *skyline* computation can be achieved in environments with specific datasets or if value normalization can be an option. For example, the properties of the sort-based SaLSa **[42]** algorithm could be injected into SAS in order to sort the dataset based on a specific scoring function and use an e*arly termination* mechanism to avoid scanning the complete list of sorted points at the beginning of the *map* phase.

5.5.2. The case of the SRSAS algorithm

In the following section, we will discuss the case of the SRSAS algorithm. The experiments for the SRSAS algorithms were performed on the distributed cluster since it is a computationally and data intensive query and can fully exploit its capabilities. Moreover, we omitted from our experiments the 1M dataset, since its cardinality is small to give insights in a distributed environment. In addition, we omitted the 2.7B dataset in order to maintain the soundness of our results by avoiding abnormal behaviours as with the 2.7B anti-correlated dataset mentioned previously. Moreover, the query point was selected randomly following the distribution of each dataset.

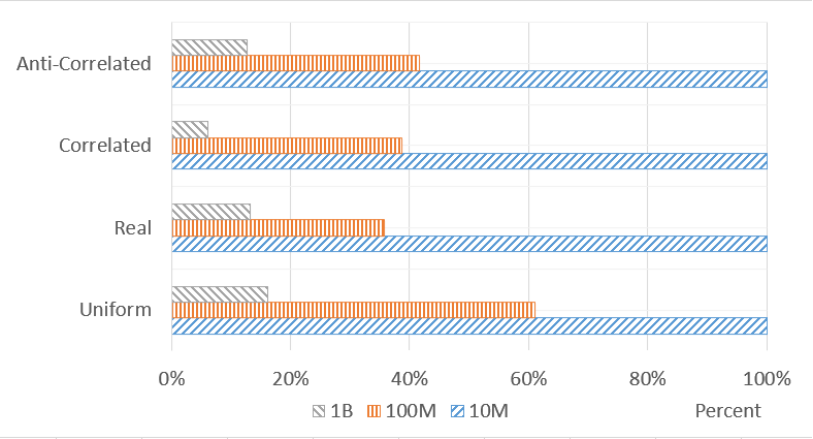


FIGURE 62: TOTAL NUMBER OF POINTS IN THE OUTPUT OF CELLFILTER AS A PERCENT OF THE INITIAL DATASET.

As presented in **Figure 62**, the *CellFilter* function in the SRSAS algorithm is more data demanding than the *SAS* (**Figure 55**). In the case of the *reverse skyline,* the algorithm needs to retrieve all the partitions that may contain *global skyline* points and that includes all the partitions that are intersected by the axis created by the query point. In small datasets, with a small number of partitions, this is equivalent to the whole dataset as with the case of the 10M dataset that has 6 partitions in total. Nevertheless, as the total number of partition in the dataset increases the number of partitions needed to be processed declines. In all the sampled versions of our datasets, the most demanding case is the one of the *uniform* dataset, which specifically for the 100M case needs on average 30% more data. Nevertheless, as the dataset size and partition number increases the pruning power of the CellFilter function increases. This can be seen in the 1B datasets where on average 12% of the dataset is needed.

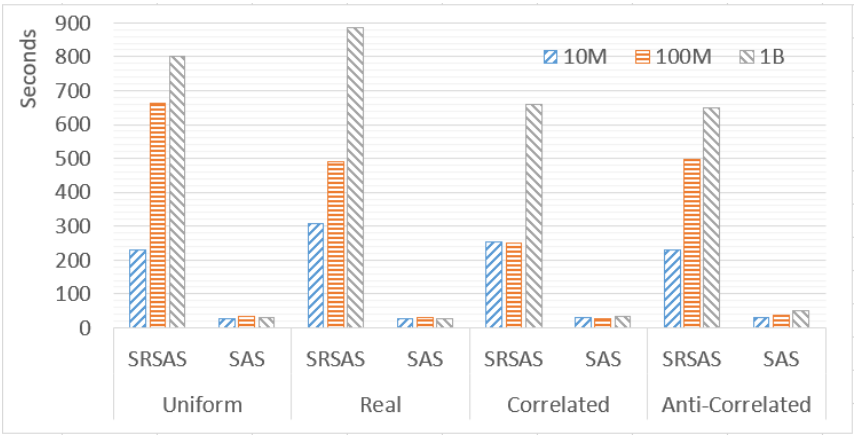


FIGURE 63: EXECUTION TIME OF SAS AND SRSAS IN DISTRIBUTED MODE OVER ALL DATASETS.

The *time cost* to compute the *reverse skyline* over the various datasets is presented in **Figure 63.** For comparison, we also provide the *time cost* to compute the *min-min skyline*. As presented, the *time cost* to compute the SRSAS is considerably more in all cases due to the nature of the query. Additionally, in comparison to **Figure 62,** the *time cost* is not analogous to the points retrieved from the *CellFilter* function but it is susceptible to the number of *global skyline* points that are identified, as we present in the following paragraphs.

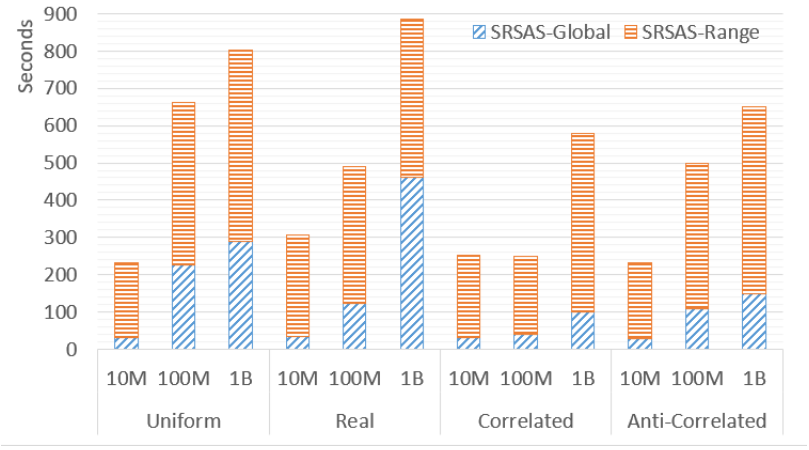


FIGURE 64: EXECUTION TIME OF SRSAS TO COMPUTE THE REVERSE SKYLINE IN DISTRIBUTED MODE OVER ALL DATASETS.

As presented in **Figure 64**, the *total time cost i*n the *reverse skyline* query can be divided in the *cost* to compute the *global skyline* and the *cost* imposed by the *range queries* needed to identify the *reverse skyline* set. In almost all cases, at least 50% of the time is consumed by the *range queries,* which reaches up to 85% in cases like the 10M *correlated* dataset concluding that the *range queries* is the major bottleneck. We remind that for each *global skyline* point, one *range query* is performed and each *range queries* may require to access data in several nodes producing multiple *map* tasks in the cluster. Since the *global skyline* set is the minimum superset of points to answer the *reverse skyline* query, the only way to effectively reduce the *time cost* is to identify overlapping regions and perform only one *range query* over each region or implement additional indexing structures.

To reduce the *cost* of *reverse skyline* computation we can set two objectives. The first is to reduce the data handled by the *map* phase, which could be achieved through a sophisticated *CellFilter* function or even with a modified indexing mechanism. The second objective could involve the creations of efficient *range query* plans or dedicated structures to avoid searching in the same area multiple times.

5.6. Conclusions and Future Work

The amount of data to be processed in order to retrieve an answer to a specific question continuously grows, while in many cases it is critical to get an answer in the minimum time. The *SpatialHadoop* framework allows using computational geometry to efficiently answer complex queries by pruning the dataset as early as possible. Furthermore, is a great framework to experiment with various queries over *MapReduce* since it supports a large number of *indexes* and *space-filling curves*.

Among many others, a family of queries that benefits from *SpatialHadoop* are the *skyline* and *reverse skyline* queries. By having an indexed dataset, we can answer consecutive queries faster in comparison to a non-indexed dataset. Since the result of a *skyline query* in a static dataset will always be the same, an indexing mechanism is more intuitive in *constrained skylines,* where the user defines the region in which the query to be performed or *reverse skylines* where the possible outcomes are infinite.

In this study, was proposed an alternative, sort-based approach to compute *skyline queries* and the *SpatialHadoop* was enhanced with *reverse skyline* queries. Intuitively, both the proposed methods can be considered as index-based and sort-based approaches simultaneously due to the *SpatialHadoop* indexing and the sorting mechanism in the *map* phase.

This work allows researchers to study similar queries such as *reverse k-skyband* and *ranked reverse queries*. Since *SpatialHadoop* is also capable of supporting *temporal* data, the *temporal skyline and temporal reverse skyline* query that were proposed in **Section 1** can be studied in conjunction with the *constrained skyline*. An alternative research direction would be to study the applicability and performance aspects of the z-order based skyline algorithm Z-SKY and the

Quad-Tree based *skyline* algorithm for *MapReduce* SKY-MR since *SpatialHadoop* also supports indexing based on *Z-order* and *Quad-Trees*.

6. SKYLINE-BASED DECISION BOUNDARY ESTIMATION

One of the most common tasks nowadays in Big Data environment is the need to classify large amount of data but based on the research conducted in **Section 3** the amount of labeled data to perform such a task is limited. There are numerous classification models, designed to perform best in different environments and datasets and each one of them has its advantages and disadvantages. However, when dealing with Big Data, their performance is significantly degraded because they are not designed or even capable of handling such large datasets. The approach proposed in this study is based on a novel proposal of exploiting the dynamics of Skyline queries to efficiently identify the decision boundary and classify Big Data. The novelty of this method is based on the fact that only small number of computations are needed in order to make a prediction, while its full potential is revealed in very large datasets.

6.1. Introduction

To deal with the problems imposed by the volume, one may consider the reduction of cardinality—or dimensionality—of the data for which various methods have been proposed. Such a problem occurs with the R-trees **[38]** when the space has more than 4–5 dimensions—named as the curse of dimensionality. The simplest case of volume reduction can be achieved by sampling techniques **[452]** that directly reduce the cardinality of the dataset. The dimensionality reduction approach can be performed either through feature elimination, extraction or selection techniques **[453, 454]**, or by directly mapping a multi-dimension dataset to a lower dimensionality space. The last approach can be performed with statistical data mining such as principle component analysis (PCA) **[455]**, stochastic approaches such as t-distributed stochastic neighbours embedding (t-SNE) **[456]** or neural network approaches such as auto encoders **[457]**.

Dimensionality reduction techniques are commonly used to improve the performance of a classifier. However, ML methods have reached a point at which we can combine even a set of weak classifiers using ensemble learning techniques **[458]** to produce good results. With this in mind, each time a new classifier is proposed, questions arise if we really need one more **[459]**.

Even with these techniques, it is not always feasible to perform a classification task with low processing costs in a big data environment, since traditional classification algorithms are designed primarily to achieve exceptional accuracy with trade-off between space or time complexity. In a k-nearest neighbours (K-NN) classifier, the main cost reflects to the cost of computing the distances from every element, in naïve Bayes to compute a large number of conditional probabilities, in probabilistic neural networks (PNN) or its radial basis function (RBF) alternative to sum local decision functions and in support vector machine (SVM) to compute complex hyperplane equations

In this work **[115, 116]**, we propose a straightforward method for classification which is significantly more efficient than traditional classification algorithms in big data environments. The proposed method uses skyline queries to identify boundary points and construct final decision boundaries. Primarily skyline queries were designed to identify the most preferable options based on certain, sometimes contradicting, optimization criteria. Skyline queries are categorized as a dominance-based multiobjective optimization approach that was developed under the scope that the dataset in use does not entirely fit in memory. The multiobjective optimization problem of efficiently identifying the skyline points has its root in the Pareto optimality problem (V. Pareto 1906) which was used in aerospace for aerodynamic shape optimization **[460]** and in economics for optimal investment portfolio **[461]**. In computational geometry, the problem is equivalent to the maximal vector problem **[462]**.

To our knowledge, this is the first work that tries to harvest the power of skyline queries in a classification process for big data. The benefits of using such an approach are:

- Even in a very large dataset, decision boundaries are described by a small number of points; thus, a classification process needs to perform only a small number of computations in order to infer the correct class;
- Decision boundaries can be independently computed, allowing for full parallelization of the whole modeling process;

- It is applicable in a wide range of multidimensional environments and specifically in any environment that its dataspace has an ordering, a feature that is inherited from the skyline query family;
- The model can be easily explained and visualized allowing for greater interpretability;
- The decision boundaries can be easily transferred, reused and easily re-optimized allowing Transfer Learning.

6.2. Methodology

As described in the previous section, there exist numerous variations and different factors on how the skyline queries can be applied on a set of data in order to extract the decision boundaries. Since this is a novel and naïve approach based on skyline search, it can be further improved and expanded. The proposed method does not rely on any specific skyline algorithms or index. To retrieve the skyline set, the BNL algorithm was chosen for its simplicity, but any other algorithm can be used. This way, the proposed method computes one skyline for each class which will eventually be part of the decision boundary construction process.

In an abstract approach, the proposed method has one preprocessing task and three normal tasks as follows:

1. Define origin points.
2. Identify Skyline points.
3. Construct decision boundaries.
4. Perform classification process

The preprocessing task deals with identifying the origin points (preferences) for which each one of the skyline queries will be performed. The first task computes the skyline based on the preferences set by the preprocessing task. The second task determines the boundaries based on the points returned by the skyline and the third one performs the classification task. Through our research we identified four different approaches on how to compute the skyline set and three different approaches on how to compute the boundaries. By using different skyline identification approaches we study how our method behaves when we retrieve a broader set of skyline points and if this assists in the estimation of the decision boundaries. Through the rest of this paper, we assume that the datasets consists of two classes (since the proposed method is binominal). As our proposed method performs best in big data environments, we targeted on a synthetic dataset of 1 M points randomly generated in space, following the Gaussian distribution. From a wide number of real datasets, we focused on a dataset that has at least 10,000 records. We note that the skyline computation is independent from the underlying distribution

6.2.1. Define the Origin Points

In order to compute the skyline set from a dataset, first we must define the preferences. It is common in literature, if not mentioned, that the minimization of preferences is desired. In our case, since we have a binominal classifier, which classifies an object between two classes, we must compute two skylines and thus we must define two origin points. In a 2-dimensional space, we have four options as preferences based on the combination of minimizing or maximizing each dimension. Each one of these points depicts one of the corners of the dataspace. Thus, in the preprocessing, we manually select which corner of the dataspace we would like to be the origin for each class, and thus, each skyline that we need to compute. This process could have been automated by taking into account the location of classes in space. Nevertheless, there should be a different approach for each case in the skyline retrieval that will be described in the following subsection. Note that the origin points that are assigned to each class are never on the same corner of dataspace. Moreover, in a 2d space there are four corner points and, in a d-dimensional space the number of required points is 2^d. Thus, for every experiment we need to perform 2^d skyline queries. The curse of dimensionality is a common issue in r-trees and skyline queries.

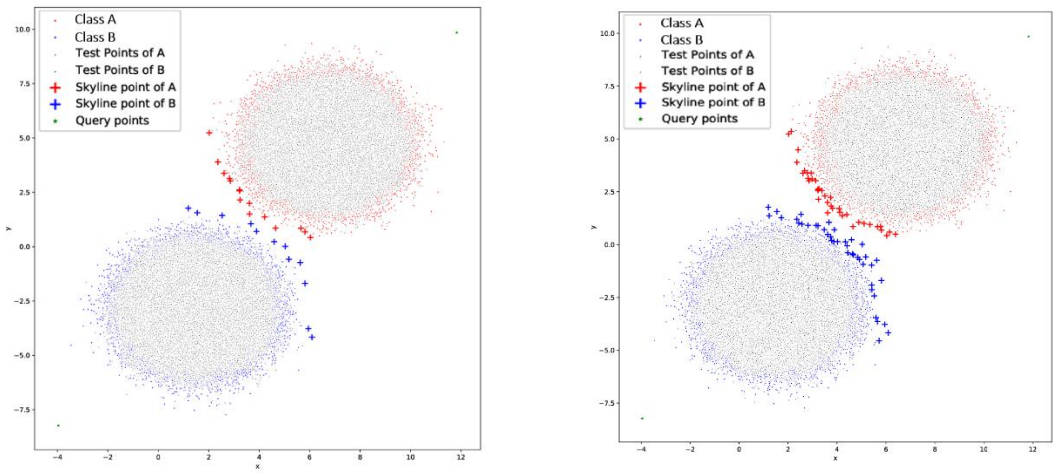
6.2.2. Identifying Skyline Points

Having defined the origin points, we can now compute the skyline for each class. This phase is completely independent for each class and thus we can parallelize the whole process. Additionally, skyline queries over Hadoop MapReduce are extendedly studied, making our method compatible with MapReduce. To accomplish this process, we studied different approaches on how we could perform one or more skyline queries in order to get a set of points that best describes a decision boundary.

Among those cases there are the single skyline which performs a single skyline for each class, the double skyline, similar to the single skyline, but it computes two skylines for each class, the opposite skyline, which tries to retrieve the skyline points that reside in two opposites sides of each class and the smart skyline which takes into account the relative location of the two classes of the dataset. Each approach has its benefits, like better accuracy, computation time and boundary approximation, but this comes to the expense of computation due to multiple skyline queries.

As previously mentioned, we use the BNL algorithm for skyline computation which has $O(kn^2)$ complexity. Thus, the complexity of identifying the points that will assist in estimating the decision boundaries is $2 * O(kn^2)$, where k is the number of dimensions and n the cardinality of each class. Below we present in detail the four different approaches for skyline computation:

1. *single skyline*: In this option, we define the origin points and perform a single skyline for each cluster, as depicted in **Figure 65 (a)**. This approach performs better in dense data as the boundaries can be straightforwardly defined. This is the simplest case with the minimum computation cost as $2 * O(kn^2)$.
2. *double skyline*: This case, as seen in **Figure 65 (b)** is similar to the one above, but it computes two skylines for each cluster using the same origins. The process computes the first skyline, removes the points from the initial dataset, but stores them in a list and then performs the second skyline computation. Then, it merges the points from the two skylines. This is done for both Cluster A and Cluster B and when completed, it proceeds to the next phase. This approach may have additional overhead as $4 * O(kn^2)$, but the resulted boundaries are more accurate in sparse data.



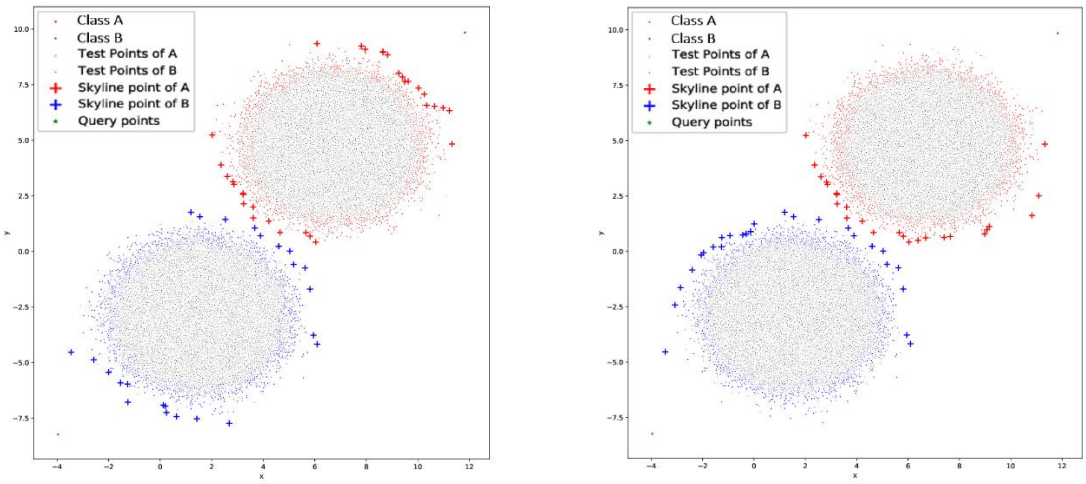
(**a**) (**b**)

FIGURE 65: (A) THE CASE OF SINGLE SKYLINE; (B) THE CASE OF DOUBLE SKYLINE

3. *opposite skyline*: The *opposite skyline*, depicted in **Figure 66 (a)** tries to retrieve the skyline points that reside in two opposite sides of each cluster. In this way, we try to enclose the area where the data on each cluster are. Even though this approach may not be desirable in many cases, it has very good results even in overlapping clusters, but it has increased

overhead as $4 * O(kn^2)$. An exception to this approach is the need of four origin points, two for each cluster which will be in the opposite side.

4. *smart skyline*: The *smart skyline* of **Figure 66 (b)** takes into account the relative location of the two clusters in order to maximize the length of the boundary line. This approach, instead of collecting more points in the same vicinity, such as the *double skyline*, retrieves points in such a way that it extends the boundary line around the cluster in order to get more chances in dividing them. This method has the same complexity as the *double* and *opposite skyline* approach.

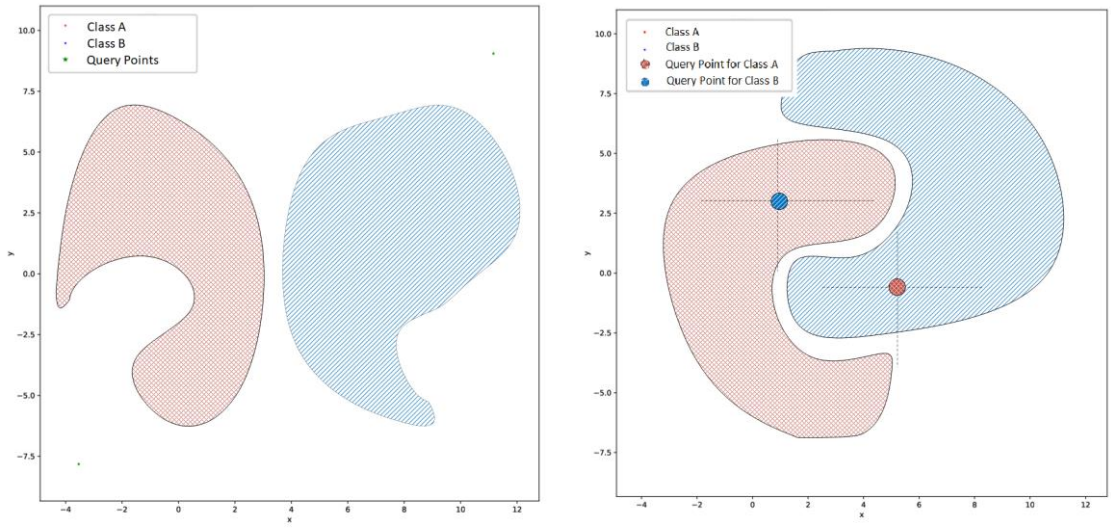


(**a**) (**b**)

FIGURE 66: (A) THE CASE OF OPPOSITE SKYLINE; (B) THE CASE OF SMART SKYLINE.

Current research on skyline computation over big data employees MapReduce-based skyline query computation approaches and have managed to compute the skyline in up to 10 dimensions and 4B records **[108]**. In such an extreme case, using a uniform distributed dataset, the number of skyline points would be approximately 3.5 M, based on $\theta\left((\ln n)^{d-1}/(d-1)!\right)$, where n is the cardinality of the dataset, d its dimensionality and θ denotes the average case scenario and is used to calculate the number of skyline points in a normal distribution **[188]**. It is common in skyline query computation that as dimensionality increases, the number of skyline points may become too numerous because the chance of one point to dominate another decreases. Taking this into account, researchers have proposed approaches to control the output size k of a skyline query and retrieve a subset of the original skyline set which holds its properties and maximizes insights. Some of those approaches are the Top-k skyline **[2, 3]**, k-representative **[62]**, Distance-based k-representative **[65]**, $\mathcal{E}$-skyline **[68]**. In highly demanding datasets where the total number k of skyline points exceeds certain thresholds, the aforementioned proposed skyline variants can be used instead of the BNL for skyline identification.

Through our experimentation, we studied the case of large scale convex shaped datasets. Reasoning about nonconvex datasets, our method can be applied in cases like the one presented in **Figure 67 (a)** following the same steps as described previously in order to define the origin points and retrieve the skyline. In more complex nonconvex datasets like a 2-class banana dataset (**Figure 67 (b)**) the identification of the skyline is more complex. In this case the two origin points that can be defined to retrieve the skyline for each class are presented in **Figure 67 (b)**. Moreover, for this case, for each origin/query point two skyline queries should be issued to properly form the decision boundaries. For the case of Class A, the two skyline queries should be issued in the upper left and lower left quadrants. For the case of Class B, the two skyline queries should be issued in the upper right and lower right quadrant. Issuing skyline queries in different quadrants can be performed by setting the appropriate preferences on minimizing or maximizing a dimension. In the case of nonconvex datasets, the variant of constrained skyline queries **[2, 3]** can be found useful in order to form partial boundaries. The case of noncontiguous datasets does not affect the skyline identification process and a single origin point can be used for each class as described in the general case.



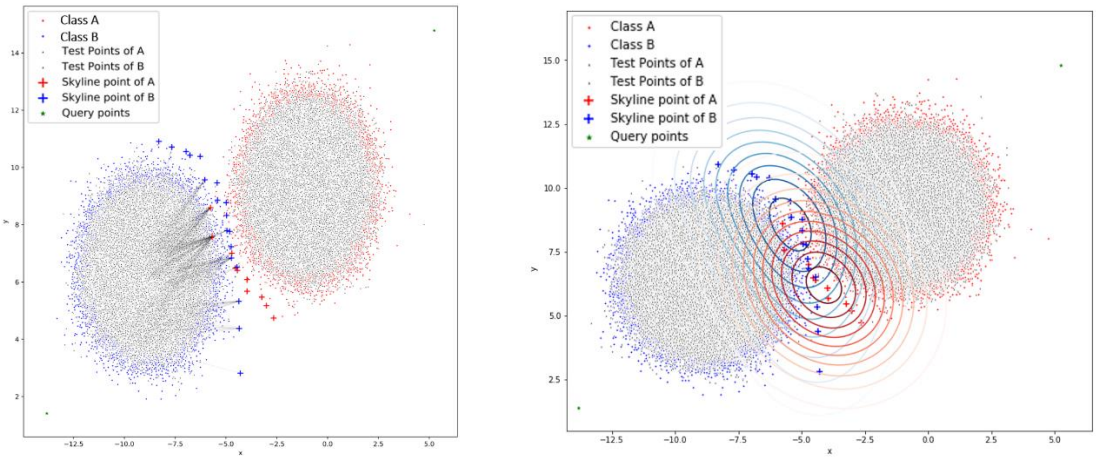
(**a**) (**b**)

FIGURE 67: (A) CONVEX DATASET; (B) BANANA DATASET.

6.2.3. Decision Boundary Construction

The boundary construction process uses the skyline points retrieved in the previous step and uses them to estimate the decision boundaries. The construction process is based on four different approaches as presented below.

1. *SKY-nearest neighbor (SKY–NN):* This is the simplest case where the decision is made based on the K-NN paradigm (**Figure 68 (a)**) by retrieving the k nearest skyline points. This method does not make any further computations to produce a boundary, but it uses the skyline points that were retrieved from the previous step as is. This approach is the easiest case to be scaled up in more than two dimensions due to the simplicity and the inherited properties of the K-NN paradigm.
2. *Parzen-window method:* The Parzen approach, visualized in **Figure 68 (b)** computes the probability that a point belongs to a certain class, based on the set of skyline points. It is a probabilistic approach that estimates a distribution or data points via a linear combination of kernels centered on the observed points of the skyline. We note that in this case only the simple skyline is used and not any variation like the probabilistic skyline.



(**a**) (**b**)

FIGURE 68: (A) THE CASE OF SKY-NEAREST NEIGHBOR (SKY-NN) APPROACH; (B) THE CASE OF PARZEN-WINDOW APPROACH.

3. *Dual curve with Polynomial Curve Fitting:* In this method, we use a curve-fitting method to compute a curve (polynomial function) **Figure 69** that best fits to our data, which in this case, are the skyline points. This case computes two curves, one for each class. A factor of importance is the degree of the polynomial function.

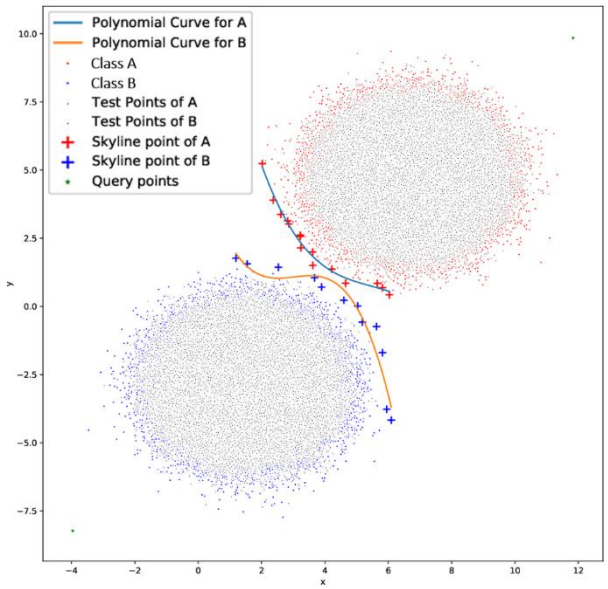
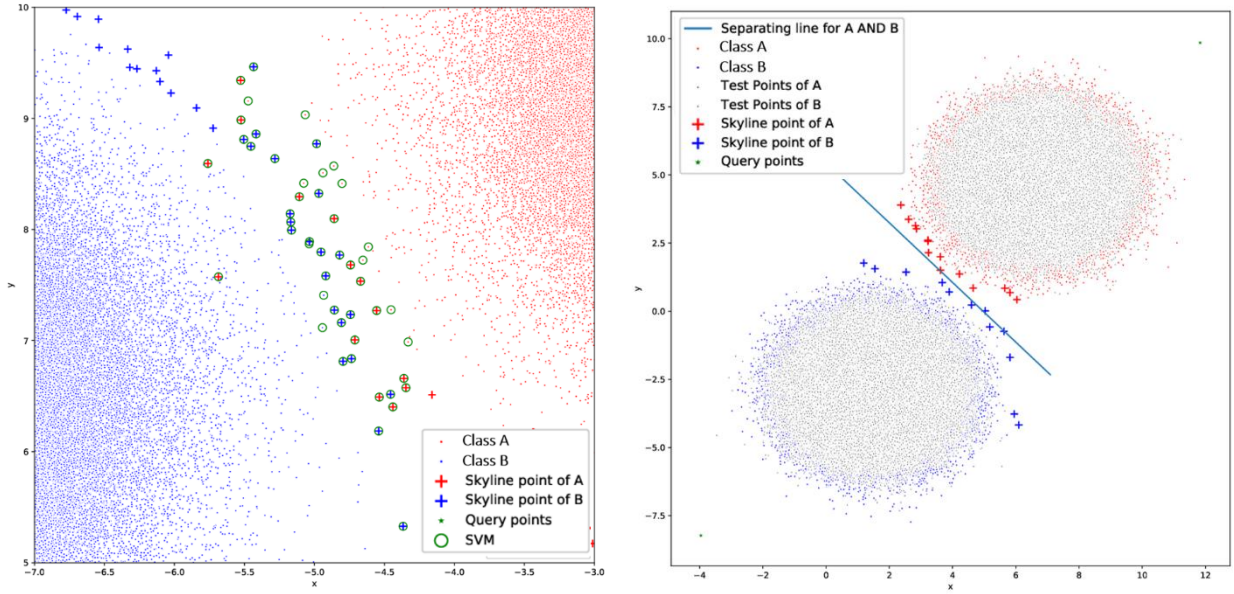


FIGURE 69: THE CASE OF POLYNOMIAL CURVE FITTING APPROACH.

4. *Single curve with Polynomial Curve Fitting:* Throughout our experimental phase, we observed that many and in some cases even all of the skyline points are a subset of the support vectors used by the final SVM (**Figure 70 (a)**). Based on this observation, this approach uses the skyline points identified from the two classes, to compute one curve or a straight line in the case presented in **Figure 70 (b)**. This final curve (line) resembles an SVM, but it is different. It can be considered as an approximate vector similar to an SVM that can be easily computed in big data environments.



(**a**) (**b**)

FIGURE 70: (A) THE SKYLINE POINTS IN COMPARISON TO THE SVM POINTS; (B) THE SEPARATING LINE PRODUCED FROM BOTH SKYLINE SETS.

6.2.4. Classification Task

After computing the decision boundaries, the classification phase starts. From the previous phase there exist two sets of points or curves that depict to the decision boundaries. During the classification process, for each point we want to classify, we compute its distance from each boundary line, or its probability based on the skyline points. The exact approach is described below.

1. *SKY-Nearest Neighbor (SKY–NN):* From each of the two sets of skyline points we retrieve the k-closest points to the point under consideration. Its class can be defined based on two alternatives, either by majority voting or by computing the total distance of the point under consideration from all the selected points of each class. The set from which the point has the smallest distance determines its class.
2. *Parzen-window method:* This method computes the probability of a point to belong to the one or the other class based on the two sets of skyline points that were retrieved. After computing the probability for the two sets of skyline points, the method assigns the point to the class with the highest probability.
3. *Dual curve with Polynomial Curve-fitting:* In this case, there exist two curves and thus two polynomial functions. Each function receives as input the x-value of the point under consideration. Both functions produce a y-value which is compared to the y-value of the point under consideration. The point will belong to the class where its y-value is closer to the one produced by the polynomial function.
4. *Single curve with Polynomial Curve-fitting:* In this case, we use the skyline points from both classes to produce a single curve and in the simplest case that we examine, a straight line. The function that represents the line receives the x-value of the point under consideration as input. It produces a y-value, which is compared to the y-value of the point. Depending on whether or not the y-value under consideration is greater or smaller than the y-value produced by the function we can infer the class that the point belongs to.

As described and presented with the previous figures, the model can be easily visualized and explained. This gives the user the ability to understand its structure and explain its output reducing the chances to produce a biased algorithm. Moreover, the boundaries can be easily transferred either in the form of a set of points or in a polynomial function and re-optimized easily in a new system if needed. Note that the computation of skylines on each class is independent and thus parallelization can be achieved. Since the proposed method is based on skyline queries, it is applicable in every environment that the skyline queries exist, like the text dataspace, which has an alphanumerical order.

The most important fact is that the skyline queries produce a small number of points, relative to the original dataset thus, the proposed method uses a very small number of points from the original dataset to estimate the boundaries. More precisely, in a 2 M dataset with two balanced classes our approach needs 14 points for each, which equals to 28 points in total. The small number of points needed to define the boundaries consecutively leads to a small number of computations during the identification of the class of a new point.

6.3. Experiments

This section presents the time needed to perform a classification task and the accuracy of our proposed method. Our intention is to show how this method behaves in close or overlapping clusters since in separable clusters 100% accuracy can be achieved. In this study we focused on two dimensional datasets since the case of high dimensional datasets would be greatly benefit by additionally studying if and how the representative and approximate skyline query computation approaches would assist in the computation of decision boundaries. In our experimentation, we used three synthetic and one real dataset consisting of a large number of points in order to describe how our method behaves in big data environments. Both datasets consist of two balanced classes. The synthetic datasets are randomly generated following a Gaussian distribution, the classes have a varying overlapping factor in order to demonstrate how our method performs and consists of 1 M points in total. The real dataset can be retrieved from **[463]** and has labeled data that describes if a person is female or male based on their height and weight. It

consists of 10,000 points and its classes have a high level of overlap. For a ground truth accuracy on the three datasets, we performed a classification task using python open-source tools for applying the k-NN, naïve Bayes and SVM classifiers. In the case of the SVM classifier we used a linear kernel and a C-value of 1. We selected those classifiers because they resemble the SKY–NN, Parzen and polynomial fit approaches that we follow. The proposed method is implemented in Java SE and all the experiments were performed with an Intel Core i5 with 6 GB RAM.

For the synthetic datasets that consist of 1 M points the naïve Bayes approach finished in less than a second, the SVM took several minutes (**Table 28** with time in milliseconds) and the k-NN did not finish in a reasonable time. This behavior reveals the problems that arise in a classification task due to the large number of computations needed in an environment with a large number of points. In terms of accuracy, both naïve Bayes and SVM achieved 100% accuracy as presented in **Table 29**. For the real dataset which consists of 10,000 points all algorithms were able to produce a result in a reasonable time (**Table 28** with time in milliseconds). Their accuracy is around 90% (**Table 29**) since the classes of the dataset have a high degree of overlap.

	k-NN	Naïve-Bayes	SVM
Synthetic Dataset I (in ms)	DNF	918	362,255
Synthetic Dataset II (in ms)	DNF	449	173,397
Synthetic Dataset III (in ms)	DNF	493	175,627
Real Dataset (in ms)	500	20	1500

TABLE 28: TIME NEEDED TO COMPUTE THE DECISION BOUNDARIES.

In **section 6.2** we discussed about the approaches of retrieving the skyline points in order to construct the boundaries and perform the classification process. Nevertheless, there are many fine-tune approaches on how many or which of the skyline points are needed to be used in the decision process for each point. Retrieving the k-NN skyline points requires $O(n)$ time and assuming that there are m points to be classified, the total overhead will be $m * O(n)$. The complexity of the polynomial curve-fitting approach depends on the method that is selected.

	k-NN	Naïve-Bayes	SVM
Synthetic Dataset I (in ms)	DNF	100.00	100.00
Synthetic Dataset II (in ms)	DNF	100.00	100.00
Synthetic Dataset III (in ms)	DNF	100.00	100.00
Real Dataset (in ms)	91.13	87.97	92.13

TABLE 29: ACCURACY WITH PYTHON AND R FRAMEWORK.

After we outlined the various approaches that can be followed, we present (**Table 30** and **Table 31**) the total time (in milliseconds) needed to identify the boundaries and perform a classification task, using three skyline points. For the polynomial case, the degree of the function is two. As presented the *SKY–NN* method is the fastest, while the *Parzen* performs better in larger datasets and the *polynomial* approach in smaller ones. Because the SKY–SVM is not applicable with *opposite skyline*, for continuity, the time and accuracy metrics will be presented at the end of each subsection that follows.

Skyline Mode	SKY-NN	Parzen	Polyn.
Single	2751	3999	12,438
Double	5423	6513	11,306
Opposite	4659	5180	13,048
Smart	4256	5360	15,833

TABLE 30: TOTAL TIME NEEDED ON AVERAGE TO PERFORM A CLASSIFICATION TASK ON THE SYNTHETIC DATASETS.

Skyline Mode	SKY-NN	Parzen	Polyn.
Single	57	103.56	89.6
Double	180.9	251.07	225.32
Opposite	117.87	201.65	178.9
Smart	70.16	160.58	105.3

TABLE 31: TOTAL TIME NEEDED TO PERFORM A CLASSIFICATION TASK ON THE REAL DATASET.

Taking into account the previously mentioned state-of-the-art classifiers, our method is faster than the k-NN and the SVM, but slower than the naïve Bayes. In terms of accuracy, as it will be presented, it achieves remarkable results achieving 100% accuracy in many cases. Those results are achieved at a reasonable time by using a small number of points, during the classification process, due to the skyline. This is ideal for a classification task in a big data environment allowing our approach to scale up in even bigger datasets where an k-NN or an SVM classifier may struggle to perform.

6.3.1. Synthetic Dataset I

The synthetic dataset (Figure 71) was constructed to have a large number of points in order to present how the method behaves in classifying a large dataset. The first of the synthetic datasets (Figure 71) is the one that stresses our method the most since the two classes slightly overlap.

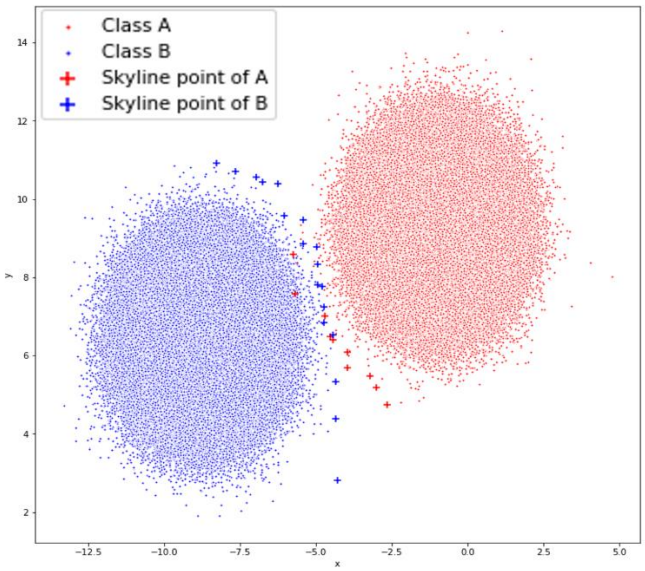


FIGURE 71: THE SINGLE SKYLINE ON THE SYNTHETIC DATASET I.

As presented in Table 32, Table 33, Table 34, Table 35, the Parzen approach outperforms the SKY–NN approach. This reveals that in this slightly overlapping scenario the distance metric that is used in the SKY–NN approach does not always infer the correct prediction in comparison to the probabilistic nature of the Parzen approach. As far as the skyline identification method the *single* (Table 32) and *double* (Table 33) *skyline* approaches perform worse than the *opposite* (Table 34) and *smart* (Table 35) *skyline*, especially when the number k of selected points is small.

Method	k					
	1	2	3	4	5	6
SKY-NN (%)	73.24	85.77	89.76	86.71	84.83	83.17

Parzen (%)	94.66	99.70	99.91	99.98	99.99	99.99

TABLE 32: SINGLE SKYLINE ON SYNTHETIC DATASET I.

Method	**k**					
	1	**2**	**3**	**4**	**5**	**6**
SKY-NN (%)	91.01	93.87	93.87	93.78	93.48	93.23
Parzen (%)	99.06	99.68	99.83	99.86	99.89	99.91

TABLE 33: DOUBLE SKYLINE ON SYNTHETIC DATASET I.

Method	**k**					
	1	**2**	**3**	**4**	**5**	**6**
SKY-NN (%)	99.99	99.99	99.99	99.99	99.99	99.99
Parzen (%)	99.94	99.96	99.98	99.99	99.99	99.99

TABLE 34: OPPOSITE SKYLINE ON SYNTHETIC DATASET I.

Method	**k**					
	1	**2**	**3**	**4**	**5**	**6**
SKY-NN (%)	99.98	99.98	99.98	99.98	99.97	99.96
Parzen (%)	99.97	99.99	99.99	100	99.99	100

TABLE 35: SMART SKYLINE ON SYNTHETIC DATASET I.

The polynomial curve-fitting approach (Table 36) shows that even with a small degree polynomial curves the method performs well, and its accuracy increases as the polynomial degree increases since it better describes the overlapping regions. From all the skyline approaches, the *single skyline* performs best, since the selected skyline points best describe the boundaries.

Degree	**Single Skyline**	**Double Skyline**	**Opposite Skylines**	**Smart Skylines**
2-nd	95.42	99.87	94.44	82.50
3-rd	98.57	60.60	99.82	98.39
5-th	99.92	99.86	99.72	97.99
7-th	99.92	99.33	99.81	93.49
8-th	99.95	99.86	99.91	99.90
11-th	99.99	99.97	99.93	99.92

TABLE 36: POLYNOMIAL CURVE FITTING ON SYNTHETIC DATASET I.

The Table 37 shows that the SKY–SVM approach is very accurate and performs better than the polynomial curve-fitting approach. Despite that the SKY–SVM is slower that the SKY–NN and Parzen approach.

Method	**Single Skyline**	**Double Skyline**	**Smart Skylines**
Accuracy	99.99	99.92	97.65
Time	7270	8068	8182

TABLE 37: SKY-SVM ON SYNTHETIC DATASET I.

6.3.2. Synthetic Dataset II

The second synthetic dataset (Figure 72) is an easier case than the previous one for our method, since the classes are very close, but not overlapping. In the *single skyline* (Table 38) the probabilistic approaches perform better. The *double skyline* (Table 39), which defines the boundaries better by using more points, works considerably better than all the methods, while the methods on Table 40 and Table 41 perform very well even with a small k value

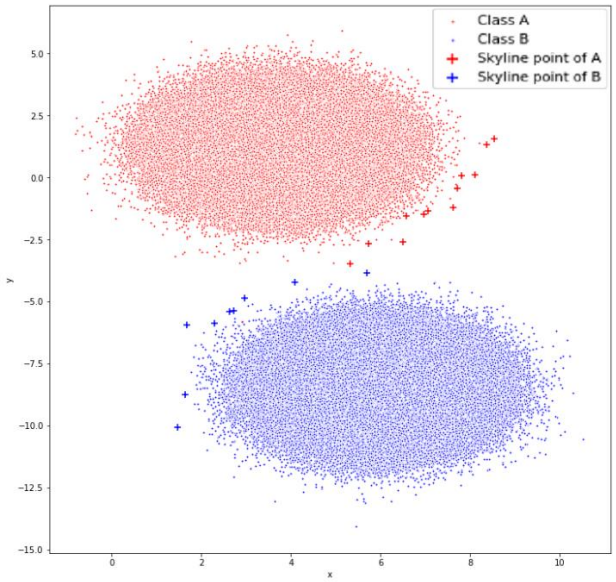


FIGURE 72: THE SINGLE SKYLINE ON THE SYNTHETIC DATASET II.

Method	k					
	1	2	3	4	5	6
SKY-NN (%)	99.98	99.96	99.87	99.82	99.75	99.69
Parzen (%)	100	100	100	100	100	100

TABLE 38: SINGLE SKYLINE ON SYNTHETIC DATASET II

Method	K					
	1	2	3	4	5	6
SKY-NN (%)	100	100	99.99	99.99	99.98	99.97
Parzen (%)	100	100	100	100	100	100

TABLE 39: DOUBLE SKYLINE ON SYNTHETIC DATASET II.

Method	k					
	1	2	3	4	5	6
SKY-NN (%)	100	100	100	100	100	100
Parzen (%)	100	100	100	100	100	100

TABLE 40: OPPOSITE SKYLINE ON SYNTHETIC DATASET II.

Method	k					
	1	2	3	4	5	6
SKY-NN (%)	100	100	100	100	100	100
Parzen (%)	100	100	100	100	100	100

TABLE 41: SMART SKYLINE ON SYNTHETIC DATASET II.

This time, for the polynomial curve-fitting approach (**Table 42**), we present the cases where the minimum polynomial degree can achieve the best results. In this case, even a 2nd degree polynomial achieves 100% accuracy.

Degree	Single Skyline	Double Skyline	Opposite Skylines	Smart Skylines
2-nd	100.00	100.00	99.94	100.00
3-rd	33.62	100.00	99.44	72.88

TABLE 42: POLYNOMIAL CURVE FITTING ON SYNTHETIC DATASET II.

Table 43 presents the accuracy and the time needed by the SKY–SVM method to perform a classification task. The method performs better than in the Dataset I with slightly better time.

Method	Single Skyline	Double Skyline	Smart Skylines
Accuracy (%)	100	100	99.90
Time (ms)	7719	8779	7864

TABLE 43: SKY-SVM ON SYNTHETIC DATASET II.

6.3.3. Synthetic Dataset III

The third and last synthetic dataset Figure 73) is the easiest case for our method, since the classes have a degree of clearance between them in such a way that a straight line could easily distinguish them. In this case all the approaches (Table 44, Table 45, Table 46, Table 47) and especially the one of Table 44 and Table 45 perform very well.

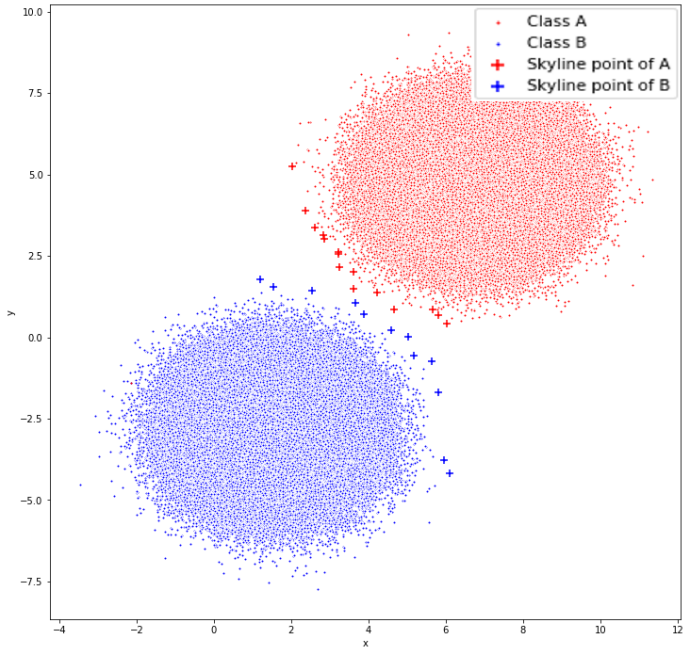


FIGURE 73: THE DATASET III.

Method	k					
	1	2	3	4	5	6
SKY-NN (%)	99.95	99.91	99.96	99.97	99.98	99.97
Parzen (%)	99.68	99.79	99.86	99.93	99.97	99.98

TABLE 44: SINGLE SKYLINE ON SYNTHETIC DATASET III.

Method	k					
	1	2	3	4	5	6
SKY-NN (%)	99.99	99.98	99.99	99.99	99.99	99.99
Parzen (%)	99.84	99.85	99.86	99.86	99.86	99.88

TABLE 45: DOUBLE SKYLINE ON SYNTHETIC DATASET III.

Method	k					
	1	2	3	4	5	6
SKY-NN (%)	100	100	100	100	100	100
Parzen (%)	100	100	100	100	100	100

TABLE 46: OPPOSITE SKYLINE ON SYNTHETIC DATASET III.

Method	k					
	1	2	3	4	5	6
SKY-NN	100	100	100	100	100	100
Parzen	100	100	100	100	100	100

TABLE 47: SMART SKYLINE ON SYNTHETIC DATASET III.

Again—as with the previous approaches—the polynomial curve-fitting approach (Table 48), performs very well for a 3rd degree polynomial.

Degree	Single Skyline	Double Skyline	Opposite Skylines	Smart Skylines
2-nd	100	100	100	100

TABLE 48: POLYNOMIAL CURVE FITTING ON SYNTHETIC DATASET III.

Table 49 reveals that the SKY–SVN with the s*ingle* skyline is the fastest and more accurate between the *double* and *smart skyline*.

Method	Single Skyline	Double Skyline	Smart Skylines
Accuracy (%)	100	100	99.97
Time (ms)	7763	10019	10229

TABLE 49: SKY-SVM ON SYNTHETIC DATASET III.

6.3.4. Real Dataset

The real dataset (Figure 74) is the one that stresses our method the most since the classes have a high degree of overlap. Additionally, as it will be presented, the correlated nature of the classes also affects the performance of our methods. In this case the value of k skyline points that are selected for the classification task varies from 7 to 13, since it gives more accurate results.

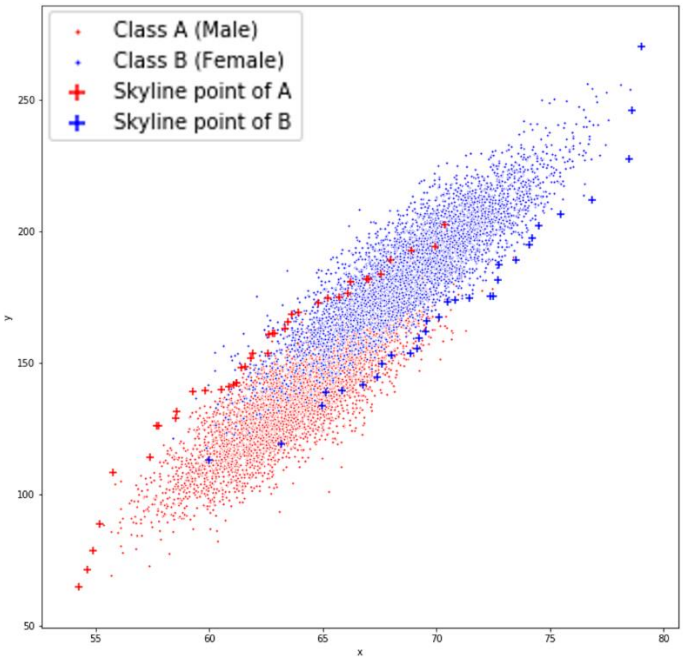


FIGURE 74: THE REAL DATASET.

As presented in Table 50, in this scenario the SKY–NN approach performs better than the Parzen approach. In addition, the SKY–NN method needs less Skyline points to infer to a correct result than the Parzen method showing that the SKY–NN method is more suitable in cases where the dataset is corelated. The *double* (Table 51) and *opposite* (Table 52) *skyline* perform worse than the *single* and the performance of the *Parzen* approach degrades significantly. This is due to the large number of skyline points produced that do not always infer the correct result. The SKY–NN method with the *smart* (Table 53) *skyline* has slightly better results than the *double* and *opposite skyline* even with a smaller number of skyline points meaning that the *smart skyline* defines the boundaries better with fewer points in this case. Overall, the *single* and *smart skyline* have comparable results on this dataset.

Method	k						
	7	8	9	10	11	12	13
SKY-NN (%)	92.40	91.80	91.30	90.83	90.87	90.80	90.93
Parzen (%)	71.67	72.73	77.27	83.20	86.37	87.40	87.67

TABLE 50: SINGLE SKYLINE ON REAL DATASET.

Method	k						
	7	8	9	10	11	12	13
SKY-NN (%)	86.66	85.96	86.1	86.4	86.3	86.3	86.1
Parzen (%)	65.66	67.2	69.13	71.8	74.36	76.76	78.56

TABLE 51: DOUBLE SKYLINE ON REAL DATASET.

Method	k						
	7	8	9	10	11	12	13
SKY-NN (%)	87.66	88.1	88	88.1	88.06	87.76	87.23
Parzen (%)	62.76	62.23	61.46	60.66	61.06	61.8	64.56

TABLE 52: OPPOSITE SKYLINE ON REAL DATASET.

Method	k						
	7	8	9	10	11	12	13
SKY-NN (%)	89.46	89.23	89.06	88.76	89	89.03	89.06
Parzen (%)	76.32	77.23	82.247	88.4	91.32	92.5	92.78

TABLE 53: SMART SKYLINE ON REAL DATASET.

The polynomial curve-fitting approach (Table 54) shows that in this case a small degree polynomial gives better results in comparison to the case of the synthetic dataset.

Degree	Single Skyline	Double Skyline	Opposite Skylines	Smart Skylines
2-nd	87	72	69	86
3-rd	82.04	68	65	81.5

TABLE 54: POLYNOMIAL CURVE FITTING ON REAL DATASET.

The accuracy of the SKY–SVM approach for the real dataset is presented in Table 55. As with the synthetic dataset the SKY–SVM approach is very accurate but takes more time to infer to a result than the SKY–NN and Parzen approach. In this case the *single* and *smart skyline* have similar results.

Method	Single Skyline	Double Skyline	Smart Skylines
Accuracy (%)	90.06	85.3	90.03

Method	Single Skyline	Double Skyline	Smart Skylines
Time (ms)	87.5	100.5	99.8

TABLE 55: SKY-SVM ON REAL DATASET.

With the use of the three synthetic datasets we present that the factor of distance between the classes does not affect the final result as opposed to the overlapping factor of classes. Based on Dataset II and III, which are not overlapping and have a variable distance between the classes, we see that all the proposed approaches can always infer to a correct result. In the case of the Dataset I for which the classes slightly overlap, we can see that the methods which are affected the most are the *single* and *double skyline* which need a larger number of k to infer to a correct result, while the *opposite* and *smart* can still infer the correct result even with a small number of k. The use of larger k-value due to the overlapping nature of classes is also presented in the case of the real dataset.

Overall, the SKY–NN approach is faster than the Parzen approach, while the number of k selected skyline points affects both methods. In addition, the SKY–NN performs better in small datasets while the Parzen in the bigger ones. In a correlated dataset, the SKY–NN performs better with fewer skyline points while the Parzen approach performs better in non-correlated datasets. The Polynomial approach works best when the polynomial function can describe the boundaries of the dataset and, in general, it is slower than the SKY–NN and Parzen method. The correlation of the dataset affects the degree of the polynomial function. The SKY–SVM approach is the slowest method and is not affected by the correlation but achieves very good results. In cases where the classes of the dataset have a high degree of overlap, we need to use more skyline points to infer in a correct result. The s*ingle skyline* approach almost in all cases achieves very good results with the least cost. The *double* and *opposite skyline* in many cases achieve very good results, but the cost is double. The *smart skyline* is a good alternative to the *single skyline* which, with small number of additional skyline points, can achieve better results.

6.4. Conclusions and Future Work

To the best of author's knowledge, this is the first work that studies the use of skyline queries in a classification process. Through our experimentation we showed that the proposed method is faster than the k-NN and the SVM, but slower than the naïve Bayes, yet having comparable accuracy when classifying large datasets. The full potential of our proposed method is visible on big datasets in which the decision boundaries are better described, and the number of points selected by the skyline operator, in comparison to the whole dataset, is small. Due to the small number of points used to describe the decision boundaries, our approach needs to perform only a small number of comparisons in order to infer the correct class. This makes our approach fast and capable of handling very large datasets for which the performance of other classifiers degrades. This was presented with the case of the K-NN approach, which did not succeed to infer in a result at a reasonable time for the case of 1 M synthetic datasets. With the use of different skyline identification methods, we showed that with a broader set of skyline points we can achieve, in some cases, better results, but with additional computation cost. Our approach can also parallelize the decision boundary computation since we can compute independently and in parallel the skyline for each class which can be beneficial when using big data technologies like Hadoop. In addition, the decision boundaries can be easily updated and optimized due to the small number of points that consists them. Moreover, due to the properties and variants of the skyline, the number of points will remain small despite the increase of the dataset size or the dimensionality. Finally, the decision boundaries can be easily visualized and interpreted by a human being allowing him to fully understand the reason for which our approach inferred to a specific result in each case.

Furthermore, our approach could be expanded by defining a metric of uncertainty in the classification process. Based on this metric the proposed method could automatically classify points and simultaneously compute a factor of uncertainty in every decision. If the uncertainty for classifying a given point drops to a certain threshold, human assistance could be requested. The user could easily identify the class and allow the system to refine the decision boundaries instantly and on the fly.

Finally, in this work we studied the case of high cardinality datasets. A future research approach is to study the case of high dimensional and high cardinality datasets which stresses the notion of skyline queries even more. In the case of high dimensional spaces in conjunction with high cardinality datasets the skyline query might return a large number of points that do not necessarily contribute, in respect with the other points of the skyline set, in defining a better decision boundary. In such a case, we can study if the approximate and representative skyline approaches can help in defining an accurate decision boundary, more efficiently, with a reduced set of skyline points.

7. CONCLUSIONS AND FUTURE DIRECTIONS

This section concludes about the research conducted in this PhD Thesis and additionally presents some future directions for research in skyline queries based on the work presented.

7.1. Conclusions

In many cases, the amount of information available and the rate of change may hide the optimal and truly desired solution. This reveals the need of a mechanism that will highlight the best options to choose among every possible scenario. Based on this the skyline query, which can be considered as a multi-objective optimization approach in database systems, was proposed. The skyline queries have a great importance in retrieving the optimal set from a given dataset, under give criteria. This mechanism is based on Pareto Optimality and retrieves the best options of a dataset by identifying the objects that present the optimal combination of the characteristics of the dataset. The research community is working in numerous topics in the field of skyline queries . As the technology advances some of these topics are Parallel and Distributed Computing, Big Data environments, Data Mining and Machine Learning which gives the opportunity of research in cutting edge research areas.

This PhD Thesis tries to give a horizontal overview on the research area of skyline queries reasoning about data, big data, big data management and supervised learning, under the spectrum of skyline queries which is a multi-objective optimization approach that can be part of decision support system. The main contribution of this Thesis is focused on five different topics:

- study the state-of-the art work on skyline queries to identify the research community trends and interests
- analyse the various data sources and the requirements that a maritime information system has
- propose a new skyline query method that considers the temporal properties of the dataset
- research new approaches in handling data over big data environments and especially in SpatialHadoop
- efficient estimate the decision boundaries in a classification process

A first step on our research approach was to study the state-of-the art work related to the skyline family and the various applications that were proposed based on numerous environments and data-specific applications. This led to the work,

- Kalyvas, C., & Tzouramanis, T. (2017). A survey of skyline query processing. arXiv preprint arXiv:1704.01788.

through which we identified the research trends that revealed the topics that are most interesting in the research community.

The next step was to study the nature of the data that exist in order to understand the overall complex nature of them and identify key areas of research that lead to the work,

- Kalyvas, C., Kokkos, A., & Tzouramanis, T. (2017). A survey of official online sources of high-quality free-of-charge geospatial data for maritime geographic information systems applications. Information Systems, 65, 36-51

through which we identified the complexity of combining data from different sources, that time and location are the common key dimensions in almost every dataset and the need of new queries and the deployment of existing ones over new environments.

From those two studies we manage to identify three research topics related to skyline queries over temporal data, big data environments and supervised learning. The first of those studies, as presented in,

- Kalyvas, C., Tzouramanis, T., & Manolopoulos, Y. (2017, April). Processing skyline queries in temporal databases. In Proceedings of the Symposium on Applied Computing (pp. 893-899).

reasons about the time variable over skyline queries and how this impacts the skyline result set. The next study reasons about the need of spatial-aware big data processing environments and how this can assist in the computation of skyline and the resource intensive reverse skyline query, as presented in

- Kalyvas, C., & Maragoudakis, M. (2019). Skyline and reverse skyline query processing in SpatialHadoop. Data & Knowledge Engineering, 122, 55-80.

Finally, through the first two studies we came across the difficulty in finding and producing labeled data that lead as to propose a new mechanism to identify decision boundaries and classify data using the properties of skyline queries, as presented in

- Kalyvas, C., & Maragoudakis, M. (2020). A Skyline-based Decision Boundary Estimation Method for Binominal Classification in Big Data. Computation, 8.3:80.
- Kalyvas, C., & Maragoudakis, M. (2020, September). A Skyline-based Decision Boundary Estimation Method for Binominal Classification in Big Data. In 2020 5th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM). IEEE.

An overview of the contribution of this Thesis is presented in **Table 56.**

Objective	Chapter	Contribution	Publication
1	2	An extensive study on the skyline queries, its variations that consists the skyline family and the various approaches that are applicable in specific environments and useful insights on open research topics that skyline queries can be applied.	[464]
2	3	An extensive study, classification, analysis of restrictions in use and distribution on the various available data sources that can be used in a maritime information system and useful insights on open research topics that skyline queries can be applied.	[451]
3	4	The proposal of a new type of skyline query named “Temporal Skyline” which integrates the dimension of time in a skyline query	[465]
4	5	The proposal of Skyline and Reverse Skyline Queries over the SpatialHadoop which encompasses indexing mechanisms in Hadoop	[108]
5	6	The proposal of a decision boundary estimation method that is based on skyline queries for binominal classification	[115, 116]

TABLE 56: OVERALL PHD THESIS CONTRIBUTIONS

Overall, the extensive study on [464] present the state of the art work related to skyline query computation. At first it presents the fundamental algorithms in skyline query computation and the basic distinction between index and non-index-based algorithms. This work also highlights the two different approaches of research in skyline queries. The first approach is related with the need to answer queries that are similar to the skyline queries and thus forming a family of skyline-related queries Some of the queries that form the skyline query family are the Constrained Skyline Queries, Dynamic Skyline Queries (DSQ), Spatial Skyline Queries (SSQ), Reverse Skyline Queries (RSQ), Group-by and Join Skyline Query, Skyline Queries Over Joins, Top-k Skyline Query, Thick Skyline Query, K-representative and Distance-based Representative Skylines, ε-Skyline, Enumerating and k-dominant Queries and the k-Skyband Queries.The second approach tries to find solutions for applying the skyline query in specific environments. Those environments can be parallel and distributed computing, wireless sensor networks and subspace skyline computation or data specific environments like skyline queries over uncertain data or metric

spaces. Moreover, this study highlights the research direction on skyline queries and assists the reader to identify the research gaps that are needed to be studied.

Due to the large research related to data-related application of skyline queries a study was conducted [451] to give a better understanding of which are the major data types that exist and identify the need to propose new queries to answer specific questions. Alongside, the restriction on use and distribution of those data was studied. Through this study was identified that time is an important aspect of data giving us the opportunity to study the skyline queries over temporal data. Furthermore, most of the datasets have a very large number of rows directing us that there is the need to study the skyline queries over the field of big data. Since we did not identify a large number of streaming data, we studied the case of skyline queries over Hadoop based environments. Finally, the problem of finding and creating label dataset in big data environments lead us to propose a skyline-based decision boundary estimation method.

As previously mentioned, the variable of time is an important aspect of data. Based on this observation the temporal skyline [465], tries to answer the skyline query taking into account time. The temporal skyline differentiates from the original BBS algorithm and uses the 3d-Rtree instead of the standard R-tree and additionally uses a modified dominance function. One example in which the temporal skyline can be very useful is the case of a hotel reservation system in which the availability and price of rooms heavily depends on the date. Through our study we identified that the results of temporal skyline queries can be quite different from the original skyline providing more useful results to the user. In addition, depending on the nature of the dataset and the existence of many unique intervals the size of the resulted query may be affected More particularly, in the extreme case where all time intervals in the dataset are distinct and non-overlapping the algorithm must traverse the entire tree and return all points. In this case Big Data processing approaches may be suitable. In the extreme case where all the time intervals are identical, the algorithm becomes the initial simple skyline query.

One of the technologies that was created to efficiently handle big data is Hadoop. With the broad adoption of Hadoop researchers continued to explore new ways to improve and expand its capabilities. One of the existing research directions is to design new technologies that incorporate well known indexing mechanisms into Hadoop. One of those technologies is SpatialHadoop which is continuously enhance with new type of queries. Based on this, in [108] we proposed an improved skyline query algorithm and enchased SpatialHadoop with reverse skyline queries. In this way we showed that we can perform skyline and reverse skyline queries over very large datasets by using the indexing capabilities of SpatiaHadoop. Through this work we identified that SpatialHadoop is a great framework to experiment with various queries over MapReduce since it supports a large number of indexes and space-filling curves, it can efficiently compute the resource demanding reverse skyline query and since it indexes the datasets can answer consecutive queries faster in comparison to cases with a non-indexed dataset.

At last, taking into account the rapid advances in data mining and machine learning we observed that skyline queries can be effectively used in a classification process. Based on the optimality of the skyline queries we used the skyline points to form the decision boundaries in a binominal classification process [115, 116]. The benefits of this approach is that the skyline set of points is small even in large datasets thus, the number of comparisons needed to infer to a result in a classification process will be small allowing the proposed method to be deployed in big data environments where the size of the dataset is very large. Moreover, the small number of points, that form the decision boundaries, makes the re-use and transferring and updating of boundaries an easy task. In overall, the SKY-NN approach is faster than the Parzen approach, while the number of k selected skyline points affects both methods. SKY-NN performs better in correlated datasets while the Parzen approach performs better in non-correlated. The SKY-SVM approach has consistent highly accurate results but is slower than the SKY-NN and Parzen approach. In addition, with high degree of overlap, more skyline points are needed with the most affected methods to be the single and double skyline. Through our study we have sawn better results with different skyline identification methods but with additional computation cost.

7.2. Future Directions

This PhD Thesis has contributed in the fields of data discovery, data management, big data processing, data mining and machine learning. Through the study of the state of the art work in the field of skyline queries [464] and the analysis of the various data sources [451] that exist, in relation to a maritime information system, we identified a large number of research topics that could be further studied. The first research topics among those involve the enhancement of the skyline query family with temporal skyline queries [465], taking into account the parameter of time which is quite of an importance in data analytics. The second one involved the identification of specialized indexing structures, like SpatialHadoop for big data processing and the proposal of algorithms for computing the skyline and the even more resource demanding reverse skyline query over large datasets [108], like the OpenStreetMap All_nodes. The third one, due to the issue of lacking large sets of labeled data, lead us to use the properties of skyline queries to build a classifier that efficiently works under big data environments and efficiently estimates decision boundary [115, 116].

In addition to the woks [465], [108], [115, 116] derived from the data analysis [451] and the state-of-the-art work in skyline queries [464] numerous potential research topics where derived. Related with Temporal skylines [465] the future work involves a study on the impact on the performance of the queries under the existence of many objects with relatively small or large time interval lifespans , the introduction of efficient algorithms for extending other skyline query variants that can be also applied to temporal data and the support of the so called why-not reverse skyline query that will aim to make a product (time-varying query point) interesting to a customer (time-varying why-not point) by modifying the product features (query attributes) and/or the customer preferences.

With regards the work on SpatialHadoop, a future study would involve the proposal of algorithms for similar queries such as reverse k-skyband and ranked reverse queries over SpatialHadoop. In addition, since SpatialHadoop is also capable of supporting temporal data, the temporal skyline and temporal reverse skyline query can be studied. Moreover, a future study would involve the applicability and performance aspects of the z-order based skyline algorithm Z-SKY and the Quad-Tree based skyline algorithm in comparison to MapReduce SKY-MR. Finally, it can be studied if ANN (Approximate Nearest Neighbor) mapreduced-based approaches are useful in identifying skyline queries in SpatialHadoop and Hadoop in general.

With regards to the work related with the decision boundary estimation through the use of skyline queries a future work involves the expansion of the proposed approach by defining a metric of uncertainty in the classification process. If the uncertainty for classifying a given point drops to a certain threshold, human assistance could be requested. Finally, a separate study should be done to identify if and how the proposed approach can assist in training of Neural Networks.

Based on the outcomes of this PhD Thesis we identified the following major future research directions:

- *Temporal skyline over SpatialHadoop* – The temporal skyline [465] retrieves the skyline query taking into account the dimension of time. Nevertheless, if the time intervals on the dataset are distinct the temporal skyline will return a large part of the dataset. This can be computationally intensive in very large datasets. Since SpatiaHadoop development is working to add the time parameter in their indexing mechanism a great future research direction would be to enhance SpatialHadoop with temporal skyline queries.
- *Skyline queries and neural networks* - Since neural networks have gained attentions again the attention of the research community after the AI winter, a good research approach would be to investigate the detection of skyline points through the use of neural networks. Such an approach could infer with high probability if a point belongs to the skyline set. Similar approaches [466] try to reduce the search space by using skyline filters based on neural networks.

- *Intelligent big data management* – The latest trends on database research and machine learning have come up with new database systems **[467]** that take into account the distribution of data. A core part of this implementation is the use of learned sorting algorithms **[468]**. Since there are numerous skyline algorithms that rely on the sorting of the dataset to identify the skyline an interesting research direction would be to study the use of this sorting algorithm in skyline query computation.
- *Data representation* – Through our research on the data sources we identified that storing, managing and processing such vast information is a complicated task. One trade-off between the information that those datasets provide in total is accuracy over time. A primary goal would be to study how accurate we want our data to be in different real-life applications and if the accuracy we want depends on time. Then, through sampling or pattern extraction techniques we can maintain a fair trade between accuracy and dataset size for each real-life application. The reduced dataset size will allow us to makes computations, predictions or pattern identification faster allowing us to find better solutions in less time.
- *Decision Optimization* - Skyline queries are a multi-objective optimization approach that retrieves the optimal solutions over a large number of possible solutions. In many cases the objectives are contradicting which is usual in decision making. In addition, there are a lot cases that decisions are made under biased environments. The use of skyline queries can assist in identifying the optimal solutions in order to make a more precise and bias-free decision.

BIBLIOGRAPHY

[1] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: Nsga-ii,” *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.
[2] 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*, 2003, pp. 467–478.
[3]Papadias, D., Tao, Y., Fu, G., and Seeger, B. , “Progressive skyline computation in database systems,” *ACM Transactions on Database Systems (TODS)*, vol. 30, no. 1, pp. 41–82, 2005.
[4] F. P. Preparata and M. I. Shamos, *Computational geometry: an introduction*. Springer Science & Business Media, 2012.
[5] H.-T. Kung, F. Luccio, and F. P. Preparata, “On finding the maxima of a set of vectors,” *Journal of the ACM (JACM)*, vol. 22, no. 4, pp. 469–476, 1975.
[6] J. L. Bentley, H.-T. Kung, M. Schkolnick, and C. D. Thompson, “On the average number of maxima in a set of vectors and applications,” *Journal of the ACM (JACM)*, vol. 25, no. 4, pp. 536–543, 1978.
[7] O. Barndorff-Nielsen and M. Sobel, “On the distribution of the number of admissible points in a vector random sample,” *Theory of Probability & Its Applications*, vol. 11, no. 2, pp. 249–269, 1966.
[8] S. Borzsony, D. Kossmann, and K. Stocker, “The skyline operator,” in *Proceedings 17th international conference on data engineering*. IEEE, 2001, pp. 421–430.
[9] J. L. Bentley, K. L. Clarkson, and D. B. Levine, “Fast linear expected-time algorithms for computing maxima and convex hulls,” *Algorithmica*, vol. 9, no. 2, pp. 168–183, 1993.
[10] P. Godfrey, R. Shipley, J. Gryz *et al.*, “Maximal vector computation in large data sets,” in *VLDB*, vol. 5, 2005, pp. 229–240.
[11] J. L. Bentley, “Multidimensional divide-and-conquer,” *Communications of the ACM*, vol. 23, no. 4, pp. 214–229, 1980.
[12] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, “When is β€□nearest neighborβ€□ meaningful?” in *International conference on database theory*. Springer, 1999, pp. 217–235.
[13] R. E. Bellman, *Adaptive control processes: a guided tour*. Princeton university press, 2015, vol. 2045.
[14] E. Dellis and B. Seeger, “Efficient computation of reverse skyline queries.” in *VLDB*, vol. 7, 2007, pp. 291–302.
[15] C. Li, B. C. Ooi, A. K. Tung, and S. Wang, “Dada: a data cube for dominant relationship analysis,” in *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, 2006, pp. 659–670.
[16] X. Lin, Y. Yuan, W. Wang, and H. Lu, “Stabbing the sky: Efficient skyline computation over sliding windows,” in *21st International Conference on Data Engineering (ICDE’05)*. IEEE, 2005, pp. 502–513.
[17] X. Huang and C. S. Jensen, “In-route skyline querying for location-based services,” in *Proceedings of the 4th international conference on Web and Wireless Geographical Information Systems*, ser. W2GIS’04. Springer-Verlag, 2005, pp. 120–135.
[18] H.-P. Kriegel, M. Renz, and M. Schubert, “Route skyline queries: A multi-preference path planning approach,” in *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*. IEEE, 2010, pp. 261–272.
[19] M. Alrifai, D. Skoutas, and T. Risse, “Selecting skyline services for qos-based web service composition,” in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 11–20.
[20] 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.
[21] B.-C. Chen, K. LeFevre, and R. Ramakrishnan, “Privacy skyline: Privacy with multidimensional adversarial knowledge,” University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2007.
[22] X. Lin, J. Xu, and H. Hu, “Authentication of location-based skyline queries,” in *Proceedings of the 20th ACM international conference on Information and knowledge management*, 2011, pp. 1583–1588.
[23] L. Chen and X. Lian, “Dynamic skyline queries in metric spaces,” in *Proceedings of the 11th international conference on Extending database technology: Advances in database technology*. ACM, 2008, pp. 333–343.
[24] C.-Y. Chan, P.-K. Eng, and K.-L. Tan, “Stratified computation of skylines with partially-ordered domains,” in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, 2005, pp. 203–214.
[25] M. E. Khalefa, M. F. Mokbel, and J. J. Levandoski, “Skyline query processing for incomplete data,” in *2008 IEEE 24th International Conference on Data Engineering*. IEEE, 2008, pp. 556–565.
[26] 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*. Citeseer, 2007, pp. 15–26.
[27] B. Jiang, J. Pei, X. Lin, and Y. Yuan, “Probabilistic skylines on uncertain data: model and bounding-pruning-refining methods,” *Journal of Intelligent Information Systems*, vol. 38, no. 1, pp. 1–39, 2012.
[28] I. F. Ilyas, G. Beskales, and M. A. Soliman, “A survey of top-k query processing techniques in relational database systems,” *ACM Computing Surveys (CSUR)*, vol. 40, no. 4, pp. 1–58, 2008.
[29] A. N. Papadopoulos and Y. Manolopoulos, *Nearest Neighbor Search:: A Database Perspective*. Springer Science & Business Media, 2006.
[30] C. Böhm and H.-P. Kriegel, “Determining the convex hull in large multidimensional databases,” in *International Conference on Data Warehousing and Knowledge Discovery*. Springer, 2001, pp. 294–306.
[31] A. Eldawy and M. F. Mokbel, “Spatialhadoop: A mapreduce framework for spatial data,” in *2015 IEEE 31st international conference on Data Engineering*. IEEE, 2015, pp. 1352–1363.
[32] D. Pertesis and C. Doulkeridis, “Efficient skyline query processing in spatialhadoop,” *Information Systems*, vol. 54, pp. 325–335, 2015.

[33] D. Borthakur, “The hadoop distributed file system: Architecture and design,” *Hadoop Project Website*, vol. 11, no. 2007, p. 21, 2007.
[34] A. Guttman, “R-trees: A dynamic index structure for spatial searching,” in *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, 1984, pp. 47–57.
[35] K.-L. Tan, P.-K. Eng, B. C. Ooi *et al.*, “Efficient progressive skyline computation,” in *VLDB*, vol. 1, 2001, pp. 301–310.
[36] R. Fagin, A. Lotem, and M. Naor, “Optimal aggregation algorithms for middleware,” *Journal of computer and system sciences*, vol. 66, no. 4, pp. 614–656, 2003.
[37] D. Kossmann, F. Ramsak, and S. Rost, “Shooting stars in the sky: An online algorithm for skyline queries,” in *VLDB’02: Proceedings of the 28th International Conference on Very Large Databases*. Elsevier, 2002, pp. 275–286.
[38] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, “The r*-tree: an efficient and robust access method for points and rectangles,” in *Proceedings of the 1990 ACM SIGMOD international conference on Management of data*, 1990, pp. 322–331.
[39] N. Roussopoulos, S. Kelley, and F. Vincent, “Nearest neighbor queries,” in *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, 1995, pp. 71–79.
[40] G. R. Hjaltason and H. Samet, “Distance browsing in spatial databases,” *ACM Transactions on Database Systems (TODS)*, vol. 24, no. 2, pp. 265–318, 1999.
[41] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, “Skyline with presorting,” in *ICDE*, vol. 3, 2003, pp. 717–719.
[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*, 2006, pp. 405–414.
[43] J. M. Hellerstein, R. Avnur, A. Chou, C. Hidber, C. Olston, V. Raman, T. Roth, and P. J. Haas, “Interactive data analysis: The control project,” *Computer*, vol. 32, no. 8, pp. 51–59, 1999.
[44] P. Mishra and M. H. Eich, “Join processing in relational databases,” *ACM Computing Surveys (CSUR)*, vol. 24, no. 1, pp. 63–113, 1992.
[45] M. Zhang and R. Alhajj, “Skyline queries with constraints: Integrating skyline and traditional query operators,” *Data & Knowledge Engineering*, vol. 69, no. 1, pp. 153 – 168, 2010.
[46] M. Sharifzadeh and C. Shahabi, “The spatial skyline queries,” in *VLDB*, 2006, pp. 751–762.
[47] M. Sharifzadeh, C. Shahabi, and L. Kazemi, “Processing spatial skyline queries in both vector spaces and spatial network databases,” *ACM Transactions on Database Systems (TODS)*, vol. 34, no. 3, p. 14, 2009.
[48] M. De Berg, O. Cheong, M. Van Kreveld, and M. Overmars, *Computational geometry: algorithms and applications*. Springer, 2008.
[49] K. Deng, X. Zhou, and H. T. Shen, “Multi-source skyline query processing in road networks,” in *2007 IEEE 23rd international conference on data engineering*. IEEE, 2007, pp. 796–805.
[50] B. Zheng, K. C. Lee, and W.-C. Lee, “Location-dependent skyline query,” in *The Ninth International Conference on Mobile Data Management (mdm 2008)*. IEEE, 2008, pp. 148–155.
[51] K. Kodama, Y. Iijima, X. Guo, and Y. Ishikawa, “Skyline queries based on user locations and preferences for making location-based recommendations,” in *GIS-LBSN*, 2009, pp. 9–16.
[52] X. Guo, Y. Ishikawa, and Y. Gao, “Direction-based spatial skylines,” in *Proceedings of the Ninth ACM International Workshop on Data Engineering for Wireless and Mobile Access*. ACM, 2010, pp. 73–80.
[53] W. Son, S. won Hwang, and H.-K. Ahn, “Mssq: Manhattan spatial skyline queries,” in *SSTD*, 2011, pp. 313–329.
[54] W. Son, S.-W. Hwang, and H.-K. Ahn, “Mssq: Manhattan spatial skyline queries,” *Information Systems*, vol. 40, pp. 67–83, 2014.
[55] M. S. Islam, R. Zhou, and C. Liu, “On answering why-not questions in reverse skyline queries,” in *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. IEEE, 2013, pp. 973–984.
[56] W. Jin, M. Ester, Z. Hu, and J. Han, “The multi-relational skyline operator,” in *ICDE*, 2007, pp. 1276–1280.
[57] M. Magnani and I. Assent, “From stars to galaxies: skyline queries on aggregate data,” in *EDBT*, 2013, pp. 477–488.
[58] C. Li, N. Zhang, N. Hassan, S. Rajasekaran, and G. Das, “On skyline groups,” in *CIKM*, 2012, pp. 2119–2123.
[59] N. Zhang, C. Li, N. Hassan, S. Rajasekaran, and G. Das, “On skyline groups,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 4, pp. 942–956, 2013.
[60] W. Jin, J. Han, and M. Ester, “Mining thick skylines over large databases,” in *PKDD*, 2004, pp. 255–266.
[61] T. Zhang, R. Ramakrishnan, and M. Livny, “Birch: an efficient data clustering method for very large databases,” *ACM sigmod record*, vol. 25, no. 2, pp. 103–114, 1996.
[62] X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang, “Selecting stars: The k most representative skyline operator,” in *ICDE*, 2007, pp. 86–95.
[63] P. Flajolet and G. N. Martin, “Probabilistic counting algorithms for data base applications,” *Journal of computer and system sciences*, vol. 31, no. 2, pp. 182–209, 1985.
[64] D. S. Hochbaum, “Approximation algorithms for the set covering and vertex cover problems,” *SIAM Journal on Computing*, vol. 11, no. 3, pp. 555–556, 1982.
[65] Y. Tao, L. Ding, X. Lin, and J. Pei, “Distance-based representative skyline,” in *ICDE*, 2009, pp. 892–903.
[66] T. F. Gonzalez, “Clustering to minimize the maximum intercluster distance,” *Theoretical Computer Science*, vol. 38, no. 0, pp. 293 – 306, 1985.
[67] D. S. Hochbaum, *Approximation algorithms for NP-hard problems*. PWS Publishing Co., 1996.
[68] T. Xia, D. Zhang, and Y. Tao, “On skylining with flexible dominance relation,” in *ICDE*, 2008, pp. 1397–1399.
[69] V. Koltun and C. H. Papadimitriou, “Approximately dominating representatives,” *Theoretical Computer Science*, vol. 371, no. 3, pp. 148–154, 2007.

[70] C. S. Jensen, C. E. Dyreson, M. Böhlen, J. Clifford, R. Elmasri, S. K. Gadia, F. Grandi, P. Hayes, S. Jajodia, W. Käfer *et al.*, “The consensus glossary of temporal database conceptsβ€”february 1998 version,” in *Temporal Databases: Research and Practice*. Springer, 1998, pp. 367–405.
[71] B. Salzberg and V. J. Tsotras, “Comparison of access methods for time-evolving data,” *ACM Computing Surveys (CSUR)*, vol. 31, no. 2, pp. 158–221, 1999.
[72] G. Ozsoyoglu and R. T. Snodgrass, “Temporal and real-time databases: A survey,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 7, no. 4, pp. 513–532, 1995.
[73] D. B. Lomet and B. Salzberg, “Transaction-time databases.” 1993.
[74] T. Tzouramanis, Y. Manolopoulos, and N. Lorentzos, “Overlapping b+-trees: an implementation of a transaction time access method,” *Data & Knowledge Engineering*, vol. 29, no. 3, pp. 381–404, 1999.
[75] M. A. Nascimento and M. H. Dunham, “Indexing valid time databases via b+-trees,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 6, pp. 929–947, 1999.
[76] C. H. Goh, H. Lu, B.-C. Ooi, and K.-L. Tan, “Indexing temporal data using existing b+-trees,” *Data & Knowledge Engineering*, vol. 18, no. 2, pp. 147–165, 1996.
[77] C. P. Kolovson and M. Stonebraker, *Segment indexes: Dynamic indexing techniques for multi-dimensional interval data*. ACM, 1991, vol. 20, no. 2.
[78] R. Bliujute, C. S. Jensen, S. Saltenis, and G. Slivinskas, “Light-weight indexing of general bitemporal data,” in *Scientific and Statistical Database Management, 2000. Proceedings. 12th International Conference on*. IEEE, 2000, pp. 125–138.
[79] Y. Theodoridis, M. Vazirgiannis, and T. Sellis, “Spatio-temporal indexing for large multimedia applications,” in *Multimedia Computing and Systems, 1996., Proceedings of the Third IEEE International Conference on*. IEEE, 1996, pp. 441–448.
[80] Y. Tao and D. Papadias, “The mv3r-tree: A spatio-temporal access method for timestamp and interval queries,” in *Proceedings of Very Large Data Bases Conference (VLDB), 11-14 September, Rome*, 2001.
[81] H.-P. Kriegel, M. Pötke, and T. Seidl, “Managing intervals efficiently in object-relational databases.” in *VLDB*, 2000, pp. 407–418.
[82] B. Jiang and J. Pei, “Online interval skyline queries on time series,” in *2009 IEEE 25th International Conference on Data Engineering*. IEEE, 2009, pp. 1036–1047.
[83] Y. Morimoto and M. A. Siddique, “Skyline sets query and its extension to spatio-temporal databases,” in *International Workshop on Databases in Networked Information Systems*. Springer, 2010, pp. 317–329.
[84] W. Gropp and E. Lusk, “Using mpi-2,” in *12th European PVM/MPI Users’ Group Meeting-Recent Advances in Parallel Virtual Machine and Message Passing Interface*, 2005.
[85] J. Dean and S. Ghemawat, “Mapreduce: simplified data processing on large clusters,” *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.
[86] S. J. Kang, S. Y. Lee, and K. M. Lee, “Performance comparison of openmp, mpi, and mapreduce in practical problems,” *Advances in Multimedia*, vol. 2015, p. 7, 2015.
[87] W.-Y. Chen, Y. Song, H. Bai, C.-J. Lin, and E. Y. Chang, “Parallel spectral clustering in distributed systems,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 3, pp. 568–586, 2011.
[88] L. Dagum and R. Menon, “Openmp: an industry standard api for shared-memory programming,” *IEEE computational science and engineering*, vol. 5, no. 1, pp. 46–55, 1998.
[89] J. Nickolls, I. Buck, M. Garland, and K. Skadron, “Scalable parallel programming with cuda,” *Queue*, vol. 6, no. 2, pp. 40–53, 2008.
[90] J. E. Stone, D. Gohara, and G. Shi, “Opencl: A parallel programming standard for heterogeneous computing systems,” *Computing in science & engineering*, vol. 12, no. 3, pp. 66–73, 2010.
[91] H. Im, J. Park, and S. Park, “Parallel skyline computation on multicore architectures,” *Information Systems*, vol. 36, no. 4, pp. 808–823, 2011.
[92] S. Chester, D. Šidlauskas, I. Assent, and K. S. Bøgh, “Scalable parallelization of skyline computation for multi-core processors,” in *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*. IEEE, 2015, pp. 1083–1094.
[93] J. Lee and S.-W. Hwang, “Scalable skyline computation using a balanced pivot selection technique,” *Information Systems*, vol. 39, pp. 1–21, 2014.
[94] S. Liknes, A. Vlachou, C. Doulkeridis, and K. Nørvåg, “Apskyline: Improved skyline computation for multicore architectures.” in *DASFAA (1)*, 2014, pp. 312–326.
[95] A. Vlachou, C. Doulkeridis, and Y. Kotidis, “Angle-based space partitioning for efficient parallel skyline computation,” in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, 2008, pp. 227–238.
[96] H. Köhler, J. Yang, and X. Zhou, “Efficient parallel skyline processing using hyperplane projections,” in *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. ACM, 2011, pp. 85–96.
[97] F. N. Afrati, P. Koutris, D. Suciu, and J. D. Ullman, “Parallel skyline queries,” *Theory of Computing Systems*, vol. 57, no. 4, pp. 1008–1037, 2015.
[98] 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*. ACM, 2013, p. 5.
[99] W. Choi, L. Liu, and B. Yu, “Multi-criteria decision making with skyline computation,” in *Information Reuse and Integration (IRI), 2012 IEEE 13th International Conference on*. IEEE, 2012, pp. 316–323.
[100] 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, 2016.

[101] B. Zhang, S. Zhou, and J. Guan, “Adapting skyline computation to the mapreduce framework: Algorithms and experiments,” *Database Systems for Adanced Applications*, pp. 403–414, 2011.
[102] L. Chen, K. Hwang, and J. Wu, “Mapreduce skyline query processing with a new angular partitioning approach,” in *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*. IEEE, 2012, pp. 2262–2270.
[103] Y. Tao, W. Lin, and X. Xiao, “Minimal mapreduce algorithms,” in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. ACM, 2013, pp. 529–540.
[104] Y. Park, J.-K. Min, and K. Shim, “Parallel computation of skyline and reverse skyline queries using mapreduce,” *Proceedings of the VLDB Endowment*, vol. 6, no. 14, pp. 2002–2013, 2013.
[105] K. Mullesgaard, J. L. Pedersen, H. Lu, and Y. Zhou, “Efficient skyline computation in mapreduce,” in *17th International Conference on Extending Database Technology (EDBT)*, 2014, pp. 37–48.
[106] J. Zhang, X. Jiang, W.-S. Ku, and X. Qin, “Efficient parallel skyline evaluation using mapreduce,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 7, pp. 1996–2009, 2016.
[107] H.-C. Ryu and S. Jung, “Mapreduce-based skyline query processing scheme using adaptive two-level grids,” *Cluster Computing*, pp. 1–12, 2017.
[108] C. Kalyvas and M. Maragoudakis, “Skyline and reverse skyline query processing in spatialhadoop,” *Data & Knowledge Engineering*, vol. 122, pp. 55–80, 2019.
[109] Z. Huang, Y. Xiang, B. Zhang, and X. Liu, “A clustering based approach for skyline diversity,” *Expert Systems with Applications*, vol. 38, no. 7, pp. 7984–7993, 2011.
[110] T. Özyer, M. Zhang, and R. Alhajj, “Integrating multi-objective genetic algorithm based clustering and data partitioning for skyline computation,” *Applied Intelligence*, vol. 35, no. 1, pp. 110–122, 2011.
[111] 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, 2020.
[112] W. Zheng, X. Lian, L. Zou, L. Hong, and D. Zhao, “Online subgraph skyline analysis over knowledge graphs,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 7, pp. 1805–1819, 2016.
[113] S. F. Nimmy, M. S. Kamal, M. I. Hossain, N. Dey, A. S. Ashour, and F. Shi, “Neural skyline filtering for imbalance features classification,” *International Journal of Computational Intelligence and Applications*, vol. 16, no. 03, p. 1750019, 2017.
[114] A. Alem, Y. Dahmani, and B. Mebarek, “Skyline computation for improving nave bayesian classifier in intrusion detection system,” *Journal homepage: http://iieta. org/journals/isi*, vol. 24, no. 5, pp. 513–518, 2019.
[115] C. Kalyvas and M. Maragkoudakis, “A skyline-based decision boundary estimation method for binominal classification in big data,” in *2020 5th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM), accepted*. IEEE, 2020.
[116] C. Kalyvas and M. Maragkoudakis , “A skyline-based decision boundary estimation method for binominal classification in big data,” *Computation*, vol. 8, no. 3, p. 80, 2020.
[117] J. Pei, W. Jin, M. Ester, and Y. Tao, “Catching the best views of skyline: A semantic approach based on decisive subspaces,” in *VLDB*, 2005, pp. 253–264.
[118] S. Agarwal, R. Agrawal, P. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan, and S. Sarawagi, “On the computation of multidimensional aggregates,” in *VLDB*, 1996, pp. 506–521.
[119] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh, “Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub totals,” *Data Min. Knowl. Discov.*, vol. 1, no. 1, pp. 29–53, 1997.
[120] J. Pei, Y. Yuan, X. Lin, W. Jin, M. Ester, Q. Liu, W. Wang, Y. Tao, J. X. Yu, and Q. Zhang, “Towards multidimensional subspace skyline analysis,” *ACM Trans. Database Syst.*, vol. 31, no. 4, pp. 1335–1381, 2006.
[121] T. Xia and D. Zhang, “Refreshing the sky: the compressed skycube with efficient support for frequent updates,” in *SIGMOD Conference*, 2006, pp. 491–502.
[122] T. Xia, D. Zhang, Z. Fang, C. X. Chen, and J. Wang, “Online subspace skyline query processing using the compressed skycube,” *ACM Trans. Database Syst.*, vol. 37, no. 2, p. 15, 2012.
[123] J. Pei, A. W.-C. Fu, X. Lin, and H. Wang, “Computing compressed multidimensional skyline cubes efficiently,” in *ICDE*, 2007, pp. 96–105.
[124] Y. Tao, X. Xiao, and J. Pei, “Subsky: Efficient computation of skylines in subspaces,” in *ICDE*, 2006, p. 65.
[125]Tao, Y., Xiao, X., and Pei, J. , “Efficient skyline and top-k retrieval in subspaces,” *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 8, pp. 1072–1088, 2007.
[126] C. Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang, “On high dimensional skylines,” in *EDBT*, 2006, pp. 478–495.
[127]Chan, C. Y., Jagadish, H. V., Tan, K. L., Tung, A K., and Zhang, Z. , “Finding k-dominant skylines in high dimensional space,” in *SIGMOD Conference*, 2006, pp. 503–514.
[128] K. C. K. Lee, B. Zheng, H. Li, and W.-C. Lee, “Approaching the skyline in z order,” in *VLDB*, 2007, pp. 279–290.
[129] V. Gaede and O. Günther, “Multidimensional access methods,” *ACM Comput. Surv.*, vol. 30, no. 2, pp. 170–231, 1998.
[130] F. Ramsak, V. Markl, R. Fenk, M. Zirkel, K. Elhardt, and R. Bayer, “Integrating the ub-tree into a database system kernel,” in *VLDB*, 2000, pp. 263–272.
[131] K. C. K. Lee, W.-C. Lee, B. Zheng, H. Li, and Y. Tian, “Z-sky: an efficient skyline query processing framework based on z-order,” *VLDB J.*, vol. 19, no. 3, pp. 333–362, 2010.
[132] M. D. Morse, J. M. Patel, and H. V. Jagadish, “Efficient skyline computation over low-cardinality domains,” in *VLDB*, 2007, pp. 267–278.

[133] S. Zhang, N. Mamoulis, and D. W. Cheung, “Scalable skyline computation using object-based space partitioning,” in *SIGMOD Conference*, 2009, pp. 483–494.
[134] J. Lee and S. won Hwang, “Bskytree: scalable skyline computation using a balanced pivot selection,” in *EDBT*, 2010, pp. 195–206.
[135] W.-T. Balke, U. Güntzer, and J. X. Zheng, “Efficient distributed skylining for web information systems,” in *EDBT*, 2004, pp. 256–273.
[136] E. Lo, K. Y. Yip, K.-I. Lin, and D. W. Cheung, “Progressive skylining over web-accessible databases,” *Data Knowl. Eng.*, vol. 57, no. 2, pp. 122–147, 2006.
[137] G. Trimponias, I. Bartolini, D. Papadias, and Y. Yang, “Skyline processing on distributed vertical decompositions,” *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 4, pp. 850–862, 2013.
[138] Z. Huang, C. S. Jensen, H. Lu, and B. C. Ooi, “Skyline queries against mobile lightweight devices in manets,” in *ICDE*, 2006, p. 66.
[139] A. Vlachou, C. Doulkeridis, Y. Kotidis, and M. Vazirgiannis, “Skypeer: Efficient subspace skyline computation over distributed data,” in *ICDE*, 2007, pp. 416–425.
[140] B. Cui, H. Lu, Q. Xu, L. Chen, Y. Dai, and Y. Zhou, “Parallel distributed processing of constrained skyline queries by filtering,” in *ICDE*, 2008, pp. 546–555.
[141] L. Chen, B. Cui, and H. Lu, “Constrained skyline query processing against distributed data sites,” *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 2, pp. 204–217, 2011.
[142] L. Zhu, Y. Tao, and S. Zhou, “Distributed skyline retrieval with low bandwidth consumption,” *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 3, pp. 384–400, 2009.
[143] S. Ratnasamy, P. Francis, M. Handley, R. M. Karp, and S. Shenker, “A scalable content-addressable network,” in *SIGCOMM*, 2001, pp. 161–172.
[144] H. V. Jagadish, B. C. Ooi, and Q. H. Vu, “Baton: A balanced tree structure for peer-to-peer networks,” in *VLDB*, 2005, pp. 661–672.
[145] P. Wu, C. Zhang, Y. Feng, B. Y. Zhao, D. Agrawal, and A. El Abbadi, “Parallelizing skyline queries for scalable distribution,” in *EDBT*, 2006, pp. 112–130.
[146] S. Wang, B. C. Ooi, A. K. H. Tung, and L. Xu, “Efficient skyline query processing on peer-to-peer networks,” in *ICDE*, 2007, pp. 1126–1135.
[147] S. Wang, Q. H. Vu, B. C. Ooi, A. K. H. Tung, and L. Xu, “Skyframe: a framework for skyline query processing in peer-to-peer systems,” *VLDB J.*, vol. 18, no. 1, pp. 345–362, 2009.
[148] L. Chen, B. Cui, H. Lu, L. Xu, and Q. Xu, “isky: Efficient and progressive skyline computing in a structured p2p network,” in *ICDCS*, 2008, pp. 160–167.
[149] B. Cui, L. Chen, L. Xu, H. Lu, G. Song, and Q. Xu, “Efficient skyline computation in structured peer-to-peer systems,” *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 7, pp. 1059–1072, 2009.
[150] B. C. Ooi, K.-L. Tan, C. Yu, and S. Bressan, “Indexing the edges - a simple and yet efficient approach to high-dimensional indexing,” in *PODS*, 2000, pp. 166–174.
[151] S. Wang, C. Yu, and B. C. Ooi, “Compressing the index - a simple and yet efficient approximation approach to high-dimensional indexing,” in *WAIM*, 2001, pp. 291–304.
[152] M. Hazewinkel, *Encyclopaedia of Mathematics*. Springer, 1993, vol. 9.
[153] D. Sacharidis, S. Papadopoulos, and D. Papadias, “Topologically sorted skylines for partially ordered domains,” in *ICDE*, 2009, pp. 1072–1083.
[154] R. C.-W. Wong, A. W.-C. Fu, J. Pei, Y. S. Ho, T. Wong, and Y. Liu, “Efficient skyline querying with variable user preferences on nominal attributes,” *PVLDB*, vol. 1, no. 1, pp. 1032–1043, 2008.
[155] R. C.-W. Wong, J. Pei, A. W.-C. Fu, and K. Wang, “Online skyline analysis with dynamic preferences on nominal attributes,” *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 1, pp. 35–49, 2009.
[156] C. Lofi, K. E. Maarry, and W. Balke, “Skyline queries in crowd-enabled databases,” in *Joint 2013 EDBT/ICDT Conferences, EDBT ’13 Proceedings, Genoa, Italy, March 18-22, 2013*, 2013, pp. 465–476.
[157] M. J. Atallah and Y. Qi, “Computing all skyline probabilities for uncertain data,” in *PODS*, 2009, pp. 279–287.
[158] M. J. Atallah, Y. Qi, and H. Yuan, “Asymptotically efficient algorithms for skyline probabilities of uncertain data,” *ACM Trans. Database Syst.*, vol. 36, no. 2, p. 12, 2011.
[159] D. Kim, H. Im, and S. Park, “Computing exact skyline probabilities for uncertain databases,” *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 12, pp. 2113–2126, 2012.
[160] X. Lian and L. Chen, “Monochromatic and bichromatic reverse skyline search over uncertain databases,” in *SIGMOD Conference*, 2008, pp. 213–226.
[161]Lian, X., and Chen, L., “Reverse skyline search in uncertain databases,” *ACM Trans. Database Syst.*, vol. 35, no. 1, 2010.
[162] X. Ding and H. Jin, “Efficient and progressive algorithms for distributed skyline queries over uncertain data,” in *ICDCS*, 2010, pp. 149–158.
[163]Ding, X., and Jin, H. , “Efficient and progressive algorithms for distributed skyline queries over uncertain data,” *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 8, pp. 1448–1462, 2012.
[164] D. Sacharidis, A. Arvanitis, and T. K. Sellis, “Probabilistic contextual skylines,” in *ICDE*, 2010, pp. 273–284.
[165] Q. Zhang, P. Ye, X. Lin, and Y. Zhang, “Skyline probability over uncertain preferences,” in *EDBT*, 2013, pp. 395–405.
[166] Y. Zhang, W. Zhang, X. Lin, B. Jiang, and J. Pei, “Ranking uncertain sky: The probabilistic top-k skyline operator,” *Inf. Syst.*, vol. 36, no. 5, pp. 898–915, 2011.
[167] C. Lofi, U. Güntzer, and W.-T. Balke, “Efficient computation of trade-off skylines,” in *EDBT*, 2010, pp. 597–608.
[168] X. Lin, Y. Zhang, W. Zhang, and M. A. Cheema, “Stochastic skyline operator,” in *ICDE*, 2011, pp. 721–732.

[169] W. Zhang, X. Lin, Y. Zhang, M. A. Cheema, and Q. Zhang, “Stochastic skylines,” *ACM Trans. Database Syst.*, vol. 37, no. 2, p. 14, 2012.
[170] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, “Models and issues in data stream systems,” in *PODS*, 2002, pp. 1–16.
[171] Y. Tao and D. Papadias, “Maintaining sliding window skylines on data streams,” *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 2, pp. 377–391, 2006.
[172] M. D. Morse, J. M. Patel, and W. I. Grosky, “Efficient continuous skyline computation,” *Inf. Sci.*, vol. 177, no. 17, pp. 3411–3437, 2007.
[173] W. W.-S. Wei, *Time series analysis*. Addison-Wesley Redwood City, California, 1994.
[174] H. Lo and G. Ghinita, “Authenticating spatial skyline queries with low communication overhead,” in *CODASPY*, 2013, pp. 177–180.
[175] B.-C. Chen, R. Ramakrishnan, and K. LeFevre, “Privacy skyline: Privacy with multidimensional adversarial knowledge,” in *VLDB*, 2007, pp. 770–781.
[176] D. J. Martin, D. Kifer, A. Machanavajjhala, J. Gehrke, and J. Y. Halpern, “Worst-case background knowledge for privacy-preserving data publishing,” in *ICDE*, 2007, pp. 126–135.
[177] S. Bothe, P. Karras, and A. Vlachou, “eskyline: processing skyline queries over encrypted data,” *Proceedings of the VLDB Endowment*, vol. 6, no. 12, pp. 1338–1341, 2013.
[178] Q. Yu and A. Bouguettaya, “Computing service skyline from uncertain qows,” *IEEE T. Services Computing*, vol. 3, no. 1, pp. 16–29, 2010.
[179] S. Wang, Q. Sun, H. Zou, and F. Yang, “Particle swarm optimization with skyline operator for fast cloud-based web service composition,” *MONET*, vol. 18, no. 1, pp. 116–121, 2013.
[180] R. Poli, J. Kennedy, and T. Blackwell, “Particle swarm optimization,” *Swarm intelligence*, vol. 1, no. 1, pp. 33–57, 2007.
[181] J. R. Munkres, “Topology,” 2000.
[182] S. Brin, “Near neighbor search in large metric spaces,” in *VLDB*, 1995, pp. 574–584.
[183] P. Ciaccia, M. Patella, and P. Zezula, “M-tree: An efficient access method for similarity search in metric spaces,” in *VLDB*, 1997, pp. 426–435.
[184] L. Chen and X. Lian, “Efficient processing of metric skyline queries,” *Knowledge and Data Engineering, IEEE Transactions on*, vol. 21, no. 3, pp. 351–365, 2009.
[185] D. Fuhry, R. Jin, and D. Zhang, “Efficient skyline computation in metric space,” in *EDBT*, 2009, pp. 1042–1051.
[186] P. Godfrey, “Skyline cardinality for relational processing,” in *FoIKS*, 2004, pp. 78–97.
[187] S. Chaudhuri, N. N. Dalvi, and R. Kaushik, “Robust cardinality and cost estimation for skyline operator,” in *ICDE*, 2006, p. 64.
[188] Z. Zhang, Y. Yang, R. Cai, D. Papadias, and A. K. H. Tung, “Kernel-based skyline cardinality estimation,” in *SIGMOD Conference*, 2009, pp. 509–522.
[189] J.-N. Hwang, S.-R. Lay, and A. Lippman, “Nonparametric multivariate density estimation: a comparative study,” *IEEE Transactions on Signal Processing*, vol. 42, no. 10, pp. 2795–2810, 1994.
[190] C. Luo, Z. Jiang, W.-C. Hou, S. He, and Q. Zhu, “A sampling approach for skyline query cardinality estimation,” *Knowl. Inf. Syst.*, vol. 32, no. 2, pp. 281–301, 2012.
[191] P. Wu, D. Agrawal, Ö. Egecioglu, and A. El Abbadi, “Deltasky: Optimal maintenance of skyline deletions without exclusive dominance region generation,” in *ICDE*, 2007, pp. 486–495.
[192] climatechange.cs.umn.edu, “Mesoscale Eddy Tracks,” http://climatechange.cs.umn.edu/eddies/index.html, 2016, accessed 2016.08.30.
[193] emsa.europa.eu, “Long Range Identification and Tracking (LRIT) system,” http://www.emsa.europa.eu/-component/flexicontent/117.html?Itemid=118, 2016, accessed 2016.08.30.
[194]europa.eu , “SafeSeaNet: Vessel traffic monitoring in E.U. waters,” http://www.emsa.europa.eu/ssn-main.html, 2016, accessed 2016.08.30.
[195]europa.eu , “ClearSeaNet: E.U. satellite-based oil spill and vessel detection service,” http://-www.emsa.europa.eu/csn-menu.html, 2016, accessed 2016.08.30.
[196]europa.eu , “European Maritime Safety Agency,” http://www.emsa.europa.eu, 2016, accessed 2016.08.30.
[197] frontex.europa.eu, “European Border Surveillance system (EUROSUR),” http://frontex.europa.eu/intelligence/-eurosur, 2016, accessed 2016.08.30.
[198] eda.europa.eu, “Maritime Surveillance (MARSUR),” https://www.eda.europa.eu/what-we-do/activities/activities-search/maritime-surveillance-(marsur), 2016, accessed 2016.08.30.
[199]europa.eu , “European Defence Agency,” http://www.eda.europa.eu, 2016, accessed 2016.08.30.
[200] ec.europa.eu, “Common Information Sharing of the Environment (CISE),” http://ec.europa.eu/maritimeaffairs/-policy/integrated_maritime_surveillance/index_en.htm, 2016, accessed 2016.08.30.
[201] copernicus.eu, “Copernicus: Europe’s eyes on Earth,” http://www.copernicus.eu, 2016, accessed 2016.08.30.
[202] ec.europa.eu, “Directorate-General for Maritime Affairs and Fisheries,” http://ec.europa.eu/dgs/-maritimeaffairs_fisheries/, 2016, accessed 2016.08.30.
[203] maritimesurveillance.security-copernicus.eu, “Copernicus: supporting projects,” http://-maritimesurveillance.security-copernicus.eu/fp7-supporting-projects, 2016, accessed 2016.08.30.
[204] myocean.eu, “MyOcean 2,” http://www.myocean.eu, 2016, accessed 2015.08.20 [currently redirected].
[205] en.wikipedia.org, “MyOcean 2,” https://en.wikipedia.org/wiki/MyOcean, 2016, accessed 2016.08.30.
[206] gisis.imo.org, “International Maritime Organization (IMO),” http://gisis.imo.org/Public/MCI/Default.aspx, 2016, accessed 2016.08.30.
[207] pssa.imo.org, “Particularly Sensitive Sea Areas (PSSA),” http://pssa.imo.org/#/globe, 2016, accessed 2016.08.30.

[208] sjofartsverket.se, “MONALISA 2.0: Securing the chain by intelligence at sea,” http://www.sjofartsverket.se/en/-MonaLisa/MONALISA-20, 2016, accessed 2016.08.30.
[209] efficiensea.org, “EfficienSea: Efficient, Safe and Sustainable Traffic at Sea,” http://www.efficiensea.org, 2016, accessed 2016.08.30.
[210] iala-aism.org, “Baltic Sea Safety (BaSSy),” http://www.iala-aism.org/wiki/iwrap/index.php/BaSSy_Project, 2016, accessed 2016.08.30.
[211] archive.northsearegion.eu, “Safety at sea,” http://archive.northsearegion.eu/iiib/projectpresentation/details/-&tid=34&theme=2, 2016, accessed 2016.08.30.
[212] navcen.uscg.gov, “U.S. Department of Homeland Security, Navigation Center: Automatic Identification System (AIS) messages,” http://www.navcen.uscg.gov/?pageName=AISMessages, 2016, accessed 2016.08.30.
[213] en.wikipedia.org, “Wikipedia: Flag of convenience,” https://en.wikipedia.org/wiki/Flag_of_convenience, 2016, accessed 2016.08.30.
[214] G. Pallotta, M. Vespe, and K. Bryan, “Vessel Pattern Knowledge Discovery from AIS Data: A Framework for Anomaly Detection and Route Prediction,” *Entropy*, vol. 15, no. 6, pp. 2218–2245, 2013.
[215] J. Faghmous, M. Le, M. Uluyol, S. Chaterjee, and V. Kumar, “Parameter-Free Spatio-Temporal Data Mining to Catalogue Global Ocean Dynamic,” in *Proc. 13th IEEE Int. Conf. Data Mining*, 2013, pp. 151–160.
[216] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer.
[217] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*. Elsevier, 2011.
[218] marinetraffic.com, “Marine Traffic: AIS Vessel Tracking,” http://www.marinetraffic.com, 2016, accessed 2016.08.30.
[219] vesseltracker.com, “Vesseltracker: Terrestrial and Satellite AIS Tracking Service in Realtime,” http://-www.vesseltracker.com, 2016, accessed 2016.08.30.
[220] imisglobal.com, “IMIS GLOBAL: MariWeb,” http://www.imisglobal.com/mariweb, 2016, accessed 2016.08.30.
[221]imisglobal.com , “IMIS Global,” http://www.imisglobal.com/, 2016, accessed 2016.08.30.
[222] shipfinder.co, “ShipFinder: Live Marine Traffic Tracking App,” http://www.shipfinder.co, 2016, accessed 2016.08.30.
[223] fleetmon.com, “FleetMon: Live AIS Vessel Tracker,” http://www.fleetmon.com, 2016, accessed 2016.08.30.
[224] lloydslistintelligence.com, “Lloyd’s List Intelligence,” http://www.lloydslistintelligence.com, 2016, accessed 2016.08.30.
[225] vesselfinder.com, “Vessel Finder: Free AIS Ship Tracking of Marine Traffic,” http://www.vesselfinder.com, 2016, accessed 2016.08.30.
[226] aishub.net, “AIS Hub data sharing center,” http://www.aishub.net, 2016, accessed 2016.08.30.
[227] ais.exploratorium.edu, “Exploratorium: San Francisco AIS Feed,” http://ais.exploratorium.edu:80, 2016, accessed 2016.08.30.
[228] imonumber.com, “IMO Number Database,” http://www.imonumber.com, 2016, accessed: 2016.03.14 [currently redirected].
[229] shiplist.net, “Ship List,” http://www.shiplist.net, 2016, accessed: 2016.03.14 [currently redirected].
[230] shipnumber.com, “ShipNumber: Ship Number, IMO Number,” http://www.shipnumber.com, 2016, accessed 2016.08.30.
[231] equasis.org, “Equasis,” http://www.equasis.org, 2016, accessed 2016.08.30.
[232] veristar.com, “VeriSTAR: Fleet Monitoring Service,” http://www.veristar.com, 2016, accessed 2016.08.30.
[233] maritime-connector.com, “Maritime Connector,” http://maritime-connector.com, 2016, accessed 2016.08.30.
[234] grosstonnage.com, “GrossTonage,” http://grosstonnage.com, 2016, accessed 2016.03.14 [occasionally offline].
[235] protectedplanet.net, “Protected Planet: Explore Protected Areas,” http://www.protectedplanet.net, 2016, accessed 2016.08.30.
[236] iucn.org, “International union for conservation of nature (iUCN): Conservation tools,” https://www.iucn.org/-resources/conservation-tools, 2016, accessed 2016.08.30.
[237] ec.europa.eu, “European Commission: The Natura 2000 network,” http://ec.europa.eu/environment/nature/-natura2000/index_en.htm, 2016, accessed 2016.08.30.
[238] natura2000.eea.europa.eu, “Natura 2000 Network Viewer,” http://natura2000.eea.europa.eu/#, 2016, accessed 2016.08.30.
[239] cdr.eionet.europa.eu, “EIONET Repository: Natura Dataset,” http://cdr.eionet.europa.eu/gr/eu/n2000/envujeg6w, 2016, accessed 2016.08.30.
[240] globalgeopark.org, “UNESCO Global Geoparks Network,” http://www.globalgeopark.org/aboutGGN/list, 2016, accessed 2016.08.30.
[241] gbif.org, “Global Biodiversity Information Facility,” http://www.gbif.org, 2016, accessed 2016.08.30.
[242] vertnet.org, “VertNet: Distributed Databases with Backbone,” http://vertnet.org, 2016, accessed 2016.08.30.
[243] nsf.gov, “NSF: National Science Foundation,” http://www.nsf.gov, 2016, accessed 2016.08.30.
[244] manisnet.org, “MaNIS: Mammal Networked Information System,” http://manisnet.org, 2016, accessed 2016.08.30.
[245] herpnet.org, “HerpNET: Global network of herpetological collections data,” http://herpnet.org, 2016, accessed 2016.08.30.
[246] fishnet2.net, “FishNet: Fish collections around the world,” http://fishnet2.net/aboutFishNet.html, 2016, accessed 2016.08.30.
[247] ornisnet.org, “ORNIS: Ornithological data,” http://www.ornisnet.org, 2016, accessed 2016.08.30.
[248] iobis.org, “OBIS: Ocean Biogeographic Information System,” http://www.iobis.org, 2016, accessed 2016.08.30.

[249] unep-wcmc.org, “UNEP-WCMC: United Nations Environment Programme’s World Conservation Monitoring Centre,” http://www.unep-wcmc.org, 2016, accessed 2016.08.30.
[250] unep.org, “United Nations Environment Programme,” http://www.unep.org, 2016, accessed 2016.08.30.
[251] reefbase.org, “ReefBase Directory,” http://www.reefbase.org/main.aspx, 2016, accessed 2016.08.30.
[252] fishbase.org, “FishBase,” http://www.fishbase.org, 2016, accessed 2016.08.30.
[253] biodiversity.europa.eu, “BISE: Biodiversity Information System for Europe,” http://www.biodiversity.europa.eu, 2016, accessed 2016.08.30.
[254] eunis.eea.europa.eu, “EUNIS: European Nature Information System,” http://eunis.eea.europa.eu, 2016, accessed 2016.08.30.
[255] emodnet.eu, “European Marine Observation and Data Network (EMODnet): Central Portal,” http://-www.emodnet.eu, 2016, accessed 2016.08.30.
[256] emodnet-seabedhabitats.eu, “EMODnet: European Seabed Habitat Maps,” http://www.emodnet-seabedhabitats.eu, 2016, accessed 2016.08.30.
[257] worldwildlife.org, “WWF: Conservation Science Data and Tools,” http://www.worldwildlife.org/pages/conservation-science-data-and-tools, 2016, accessed 2016.08.30.
[258] iucnredlist.org, “The IUCN Red List of Threatened Species: Spatial Data,” http://www.iucnredlist.org/technical-documents/spatial-data, 2016, accessed 2016.08.30.
[259] pleiades.stoa.org, “Pleiades: Ancient Places,” http://pleiades.stoa.org/places, 2016, accessed 2016.08.30.
[260] awmc.unc.edu, “Ancient World Mapping Center,” http://awmc.unc.edu/wordpress/map-files, 2016, accessed 2016.08.30.
[261] archaeocadastre.culture.gr, “Archaeological Cadastre,” http://archaeocadastre.culture.gr/el/data, 2016, accessed 2016.08.30.
[262] ascsa.edu.gr, “The American School of Classical Studies at Athens: Maps and GIS data for Corinth and Greece,” http://www.ascsa.edu.gr/index.php/excavationcorinth/maps-and-gis-data-for-corinth-and-greece, 2016, accessed 2016.08.30.
[263] dare.ht.lu.se, “Digital Atlas of the Roman Empire,” http://dare.ht.lu.se, 2016, accessed 2016.08.30.
[264] commons.pelagios.org, “Pelagios Commons: Linked Open Geodata in the Humanities,” http://-commons.pelagios.org, 2016, accessed 2016.08.30.
[265] whc.unesco.org, “UNESCO World Heritage list,” http://whc.unesco.org/en/list, 2016, accessed 2016.08.30.
[266] machuproject.eu, “MACHU: Managing Cultural Heritage Underwater,” http://www.machuproject.eu, 2016, accessed 2016.08.30.
[267] ocean-energy-systems.org, “Ocean Energy Systems: Ocean Energy in the World,” https://www.ocean-energy-systems.org/ocean-energy-in-the-world/, 2016, accessed 2016.08.30.
[268] windfarmaction.com, “Windfarm Action Group: World Windfarm List,” http://www.windfarmaction.com/world-windfarm-lists.html, 2016, accessed 2016.08.30.
[269] cablemap.info, “Greg’s Cable Map,” http://www.cablemap.info, 2016, accessed 2016.08.30.
[270] telegeography.com, “TeleGeography: Submarine Cable Map,” https://www.telegeography.com/telecom-resources/submarine-cable-map/index.html, 2016, accessed 2016.08.30.
[271] sccl.no, “Subsea cables consultants Ltd.” http://www.sccl.no/references.php, 2016, accessed 2016.08.30.
[272] iscpc.org, “International Cable Protection Committee: Cable Data,” https://www.iscpc.org/cable-data, 2016, accessed 2016.08.30.
[273] kis-orca.eu, “KIS-ORCA: Interactive Map,” http://www.kis-orca.eu/map, 2016, accessed 2016.08.30.
[274] emodnet-humanactivities.eu, “EMODnet: Offshore Installations,” http://www.emodnet-humanactivities.eu/search-results.php?dataname=Offshore+Installations, 2016, accessed 2016.08.30.
[275] prio.org, “PRIO: Geographical and Resource Datasets,” https://www.prio.org/Data/Geographical-and-Resource-Datasets, 2016, accessed 2016.08.30.
[276] energy.usgs.gov, “USGS: World Petroleum Assessment,” http://energy.usgs.gov/OilGas/AssessmentsData/-WorldPetroleumAssessment.aspx#3882216-data, 2016, accessed 2016.08.30.
[277] theodora.com, “Countries of the World: World Pipelines maps,” http://www.theodora.com/pipelines/-world_oil_gas_and_products_pipelines.html, 2016, accessed 2016.08.30.
[278] bsu-bund.de, “Federal Bureau of Maritime Casualty Investigation in Germany,” http://www.bsu-bund.de/EN, 2016, accessed 2016.08.30.
[279] hbmci.gov.gr, “Hellenic Bureau for Marine Casualties Investigation,” http://www.hbmci.gov.gr, 2016, accessed 2016.08.30.
[280] ntsb.gov, “National Transportation Safety Board: Marine Accidents Report,” http://www.ntsb.gov/investigations/-AccidentReports/Pages/marine.aspx, 2016, accessed 2016.08.30.
[281]ntsb.gov , “National Transportation Safety Board: Marine Accidents Animation,” http://www.ntsb.gov/Pages/-animations.aspx#MS, 2016, accessed 2016.08.30.
[282] en.wikipedia.org, “Wikipedia: List of maritime disasters,” http://en.wikipedia.org/wiki/List_of_maritime_disasters, 2016, accessed 2016.08.30.
[283] wrecksite.eu, “Wrecksite database,” http://www.wrecksite.eu/, 2016, accessed 2016.08.30.
[284] maiif.org, “The Marine Accident Investigator’s International Forum (MAIFF),” http://www.maiif.org/index.php/-investigation-reports, 2016, accessed 2016.08.30.
[285] odin.tc, “Maritime Bulletin,” http://www.odin.tc/en2016/, 2016, accessed 2016.08.30.
[286] emsa.europa.eu, “European Maritime Safety Agency (EMSA),” http://www.emsa.europa.eu/implementation-tasks/accident-investigation.html, 2016, accessed 2016.08.30.

[287] itopf.com, “ITOPF: GIS Data Resources,” http://www.itopf.com/knowledge-resources/data-statistics/gis, 2016, accessed 2016.08.30.
[288] incidentnews.noaa.gov, “NOAA: IncidentNews,” http://incidentnews.noaa.gov, 2016, accessed 2016.08.30.
[289] noaa.gov, “National Oceanic and Atmospheric Administration (NOAA), U.S. Department of Commerce,” http://-www.noaa.gov, 2016, accessed 2016.08.30.
[290] response.restoration.noaa.gov, “NOAA: Office of Response and Restoration,” http://-response.restoration.noaa.gov, 2016, accessed 2016.08.30.
[291] itfglobal.org, “International Transport Workers Federation,” http://www.itfglobal.org, 2016, accessed 2016.08.30.
[292] en.wikipedia.org, “Wikipedia: List of flags of convenience,” https://en.wikipedia.org/wiki/-List_of_flags_of_convenience, 2016, accessed 2016.08.30.
[293] imo.org, “Port State Control,” http://www.imo.org/en/OurWork/MSAS/Pages/PortStateControl.aspx, 2016, accessed 2016.08.30.
[294] parismou.org, “Europe and the north Atlantic Memorandum of Understanding (Paris MoU),” https://-www.parismou.org/detentions-banning/current-detentions, 2016, accessed 2016.08.30.
[295] tokyo-mou.org, “Asia and the Pacific Memorandum of Understanding (Tokyo MoU),” http://www.tokyo-mou.org/-inspections_detentions/psc_database.php, 2016, accessed 2016.08.30.
[296] acuerdolatino.int.ar, “Latin America Memorandum of Understanding (Acuerdo de Viña del Mar),” http://-www.acuerdolatino.int.ar/ciala/index.php, 2016, accessed 2016.08.30.
[297] caribbeanmou.org, “Caribbean Memorandum of Understanding (Caribbean MoU),” http://-www.caribbeanmou.org/, 2016, accessed 2016.08.30.
[298] abujamou.org, “West and Central Africa Memorandum of Understanding (Abuja MoU),” http://www.abujamou.org/-index.php?pid=125disclaimer, 2016, accessed 2016.08.30.
[299] bsmou.org, “Black Sea Memorandum of Understanding (Black Sea MoU),” http://www.bsmou.org/database/, 2016, accessed 2016.08.30.
[300] medmouic.org, “Mediterranean Sea region Memorandum of Understanding (Mediterranean MoU),” http://-www.medmouic.org, 2016, accessed 2016.08.30.
[301] iomou.org, “Indian Ocean Memorandum of Understanding (Indian Ocean MoU),” http://www.iomou.org/-moumain.htm, 2016, accessed 2016.08.30.
[302] riyadhmou.org, “Riyadh Memorandum of Understanding (Riyadh MoU),” http://www.riyadhmou.org/-basicsearch.html?lang=en, 2016, accessed 2016.08.30.
[303] uscg.mil, “USCG Memorandum of Understandings,” https://www.uscg.mil/auxiliary/administration/mou-national.asp, 2016, accessed 2016.08.30.
[304] msi.nga.mil, “NGA Portal: Anti-shipping Activity Messages,” http://msi.nga.mil/NGAPortal/-MSI.portal?_nfpb=true&_st=&_pageLabel=msi_portal_page_65, 2016, accessed 2016.08.30.
[305] programmableweb.com, “Programmable Web,” http://www.programmableweb.com/search/weather, 2016, accessed 2016.08.30.
[306] worldweatheronline.com, “World Weather Online,” http://www.worldweatheronline.com, 2016, accessed 2016.08.30.
[307] wunderground.com, “Weather Underground,” http://www.wunderground.com/weather/api, 2016, accessed 2016.08.30.
[308] severe.worldweather.org, “Severe Weather Information Center,” http://severe.worldweather.org, 2016, accessed 2016.08.30.
[309] meteoalarm.eu, “Meteoalarm: Alerting Europe for extreme weather,” http://www.meteoalarm.eu, 2016, accessed 2016.08.30.
[310] wmo.asu.edu, “ASU World Meteorological Organization: Global Weather & Climate Extremes,” http://-wmo.asu.edu, 2016, accessed 2016.08.30.
[311] earthobservatory.nasa.gov, “NASA: Earth Observatory,” http://earthobservatory.nasa.gov/GlobalMaps/-?eocn=topnav, 2016, accessed 2016.08.30.
[312] ypeka.gr, “Hellenic Ministry of Environment, Energy and Climate Change (YPEKA): Air pollution reports,” http://-www.ypeka.gr/Default.aspx?tabid=489&language=el-GR, 2016, accessed 2016.08.30.
[313] eea.europa.eu, “European Environment Agency (EEA): Air Quality Levels in Europe,” http://www.eea.europa.eu/-themes/air/air-quality/map, 2016, accessed 2016.08.30.
[314]europa.eu , “European Environment Agency,” http://www.eea.europa.eu, 2016, accessed 2016.08.30.
[315] aws.amazon.com, “Daily Global Weather Measurements 1929-2009,” https://aws.amazon.com/datasets/daily-global-weather-measurements-1929-2009-ncdc-gsod/, 2016, accessed 2016.08.30.
[316] pendientedemigracion.ucm.es, “Climatological Database for the World’s Oceans,” http://-pendientedemigracion.ucm.es/info/cliwoc, 2016, accessed 2016.08.30.
[317] eca.knmi.nl, “European Climate Assessment & Dataset,” http://eca.knmi.nl, 2016, accessed 2016.08.30.
[318] gein.noa.gr, “Greek Institute of Geodynamics,” http://www.gein.noa.gr/el/teleutaia-anakoinothenta, 2016, accessed 2016.08.30.
[319] emsc-csem.org, “The European-Mediterranean Seismological Centre (EMSC): Real-time earthquake information,” http://www.emsc-csem.org, 2016, accessed 2016.08.30.
[320] geofon.gfz-potsdam.de, “GEOFON Program,” http://geofon.gfz-potsdam.de/mission, 2016, accessed 2016.08.30.
[321] earthquake.usgs.gov, “USGS: Earthquake Hazards Program,” http://earthquake.usgs.gov/data, 2016, accessed 2016.08.30.
[322] itic.ioc-unesco.org, “International Tsunami Information Center,” http://itic.ioc-unesco.org/-index.php?option=com_content&view=category&layout=blog&id=1164&Itemid=1164, 2016, accessed 2016.08.30.

[323] preview.grid.unep.ch, “Global Risk Data Platform On Natural Events,” http://preview.grid.unep.ch/index.php, 2016, accessed 2016.08.30.
[324] volcano.si.edu, “Volcanoes of the World,” http://volcano.si.edu, 2016, accessed 2016.08.30.
[325] msi.nga.mil, “NGA Portal: The World Port Index,” http://msi.nga.mil/NGAPortal/-MSI.portal?_nfpb=true&_pageLabel=msi_portal_page_62&pubCode=0015, 2016, accessed 2016.08.30.
[326] wikimapia.org, “Wikimapia: Describing the world,” http://wikimapia.org, 2016, accessed 2016.08.30.
[327] ourairports.com, “OurAirports: Open Data,” http://ourairports.com/data, 2016, accessed 2016.08.30.
[328] openflights.org, “OpenFlights: Airport, airline and route data,” http://openflights.org/data.html, 2016, accessed 2016.08.30.
[329] segelflug.de, “WELT2000: Database of airports and waypoints worldwide,” http://www.segelflug.de/vereine/-welt2000, 2016, accessed 2016.08.30.
[330] bluemarblegeo.com, “Blue Marble Geographics: World Data Map,” http://www.bluemarblegeo.com/products/-world-map-data-download.php, 2016, accessed 2016.08.30.
[331] msi.nga.mil, “NGA Portal: Central Mediterranean Dataset,” http://msi.nga.mil/NGAPortal/-DNC.portal?_nfpb=true&_st=&_pageLabel=dnc_portal_page_61&regionCode=09, 2016, accessed 2016.08.30.
[332] shoreline.noaa.gov, “NOAA Shoreline Website,” http://shoreline.noaa.gov, 2016, accessed 2016.08.30.
[333] ghin.pdc.org, “Global Hazards Information Network, Pacific Disaster Center,” http://ghin.pdc.org/ghin/catalog/-main/home.page, 2016, accessed 2016.08.30.
[334] soest.hawaii.edu, “A Global Self-consistent, Hierarchical, High-resolution Geography Database,” http://-www.soest.hawaii.edu/pwessel/gshhg, 2016, accessed 2016.08.30.
[335] eea.europa.eu, “EEA: Coastline for Analysis,” http://www.eea.europa.eu/data-and-maps/data/eea-coastline-for-analysis, 2016, accessed 2016.08.30.
[336]europa.eu , “EEA: Maritime Boundaries,” http://www.eea.europa.eu/data-and-maps/data/maritime-boundaries, 2016, accessed 2016.08.30.
[337] eurosion.org, “EUROSION: A European initiative for sustainable coastal erosion management,” http://-www.eurosion.org, 2016, accessed 2016.08.30.
[338] eea.europa.eu, “EEA: Coasts and Sea Datasets,” http://www.eea.europa.eu/data-and-maps/-data#c5=all&b_start=0&c9=waterbase&c0=20&c11=coast_sea, 2016, accessed 2016.08.30.
[339] earth.esa.int, “European Space Agency (ESA): IMAGE 2006,” https://earth.esa.int/web/guest/-/esa-datasets-6287, 2016, accessed 2016.08.30.
[340] ngdc.noaa.gov, “National Center for Environmental Information,” http://www.ngdc.noaa.gov, 2016, accessed 2016.08.30.
[341] maps.ngdc.noaa.gov, “NGDC’s Trackline Geophysical Data,” http://maps.ngdc.noaa.gov/viewers/geophysics, 2016, accessed 2016.08.30.
[342]noaa.gov , “NGDC’s Bathymetry Data,” http://maps.ngdc.noaa.gov/viewers/bathymetry, 2016, accessed 2016.08.30.
[343] iho.int, “International Hydrography Organization,” http://iho.int/srv1/-index.php?option=com_content&view=article&id=300&Itemid=744&lang=en, 2016, accessed 2016.08.30.
[344] gebco.net, “General Bathymetric Chart of the Oceans (GEBCO),” http://www.gebco.net, 2016, accessed 2016.08.30.
[345] bodc.ac.uk, “British Oceanographic Data Centre: GEBCO Dataset,” https://www.bodc.ac.uk/data/online_delivery/-gebco, 2016, accessed 2016.08.30.
[346]bodc.ac.uk , “British Oceanographic Data Centre,” http://www.bodc.ac.uk, 2016, accessed 2016.08.30.
[347] portal.emodnet-hydrography.eu, “EMODnet Bathymetry Portal,” http://portal.emodnet-hydrography.eu, 2016, accessed 2016.08.30.
[348] tidesandcurrents.noaa.gov, “NOAA’s Center for Operational Oceanographic Products and Services,” https://-tidesandcurrents.noaa.gov, 2016, accessed 2016.08.30.
[349]noaa.gov , “NOAA: Tides & Currents,” http://tidesandcurrents.noaa.gov/googleearth.shtml, 2016, accessed 2016.08.30.
[350] ioc-sealevelmonitoring.org, “Sea Level Station Monitoring Facility,” http://www.ioc-sealevelmonitoring.org/-map.php, 2016, accessed 2016.08.30.
[351] geoportal.org, “GEOSS Portal: Discover, Access, Contribute Earth Observations, Information and Services,” http://geoportal.org/web/guest/geo_home_stp, 2016, accessed 2016.08.30.
[352] naturalearthdata.com, “The Natural Earth: A Portal for Free Vector and Raster Map Data,” http://-www.naturalearthdata.com, 2016, accessed 2016.08.30.
[353] reverb.echo.nasa.gov, “NASA’s Earth Observation System,” http://reverb.echo.nasa.gov/reverb, 2016, accessed 2016.08.30.
[354] libguides.mit.edu, “MIT Geodata Repository,” http://libguides.mit.edu/gis/Geodata, 2016, accessed 2016.08.30.
[355] nauticalcharts.noaa.gov, “NOOA’s Office of Coast Survey,” http://www.nauticalcharts.noaa.gov, 2016, accessed 2016.08.30.
[356] gif.berkeley.edu, “Geospatial Innovation Facility,” http://gif.berkeley.edu/resources/data_subject.html, 2016, accessed 2016.08.30.
[357] eea.europa.eu, “EEA: Datasets,” http://www.eea.europa.eu/data-and-maps/data#c17=&c5=all&c0=5&b_start=0, 2016, accessed 2016.08.30.
[358] earth.esa.int, “European Space Agency,” https://earth.esa.int/web/guest/data-access/browse-data-products, 2016, accessed 2016.08.30.

[359] ec.europa.eu, “EuroStat: European Statistics,” http://ec.europa.eu/eurostat/data/database, 2016, accessed 2016.08.30.
[360] marineplan.es, “Marineplan: Maritime Policy and Marine Spatial Planning,” http://www.marineplan.es/ES/en/-google-earth, 2016, accessed 2016.08.30.
[361] marinecoastalgis.net, “Davey Jones Locker,” http://marinecoastalgis.net, 2016, accessed 2016.08.30.
[362] J. Maurer, “Atlas of the Cryosphere,” *National Snow and Ice Data Center (26 January 2012).*, 2007, accessed 2016.08.30.
[363] quantarctica.npolar.no, “Quantarctica,” http://quantarctica.npolar.no, 2016, accessed 2016.08.30.
[364] diva-gis.org, “DIVAGIS: Free GIS Data,” http://www.diva-gis.org/Data, 2016, accessed 2016.08.30.
[365] hcmr.gr, “Hellenic Centre for Marine Research (HCMR),” http://www.hcmr.gr/en/articlepage.php?id=108, 2016, accessed 2016.08.30.
[366] hnodc.hcmr.gr, “Institute of Oceanography: Hellenic Centre for Marine Research,” http://hnodc.hcmr.gr, 2016, accessed 2016.08.30.
[367] mapserver.ath.hcmr.gr, “Hellenic Centre for Marine Research (HCMR): Online Search & Download Service,” http://mapserver.ath.hcmr.gr/pagin/v27/index_new.php, 2016, accessed 2016.08.30.
[368] hnodc.ath.hcmr.gr, “European Directory of Marine Environmental Datasets,” http://hnodc.ath.hcmr.gr/edmed/-SearchData.php, 2016, accessed 2016.08.30.
[369]hcmr.gr , “EDIOS database for Eastern Mediterranean and Black Sea,” http://hnodc.ath.hcmr.gr/edios/-edios_db.html, 2016, accessed 2016.08.30.
[370] freegisdata.rtwilson.com, “FreeGISData: Free GIS Datasets,” http://freegisdata.rtwilson.com, 2016, accessed 2016.08.30.
[371] free-gis-data.blogspot.gr, “Free GIS Blogspot: Free GIS, Remote Sensing, Spatial and Hydrology Data,” http://-free-gis-data.blogspot.gr, 2016, accessed 2016.08.30.
[372] N. Arundale, “AIS Decoder: Decodes All 27 AIS message types,” http://arundale.com/docs/ais/ais_decoder.html, 2016, accessed 2016.08.30.
[373] shipais.com, “Shipais: Watching the ships go by,” http://www.shipais.com/doc/links-2.php, 2016, accessed 2016.08.30.
[374] cruiseshipportal.com, “CruiseShip Portal: AIS live Ships Tracking,” http://www.cruiseshipportal.com/categories/-other/ais-live-ship-tracking.html, 2016, accessed 2016.08.30.
[375] liveaisworld.yachtmarine.com, “LiveAISWorld,” http://liveaisworld.yachtmarine.com/LIVEAISWORLD.html, 2016, accessed 2016.08.30.
[376] geonames.nga.mil, “GNS: NGA GEOnet Names Server,” http://geonames.nga.mil/gns/html, 2016, accessed 2016.08.30.
[377] copernicus.eu, “Copernicus: Services,” http://www.copernicus.eu/main/services, 2016, accessed 2016.08.30.
[378] scihub.copernicus.eu, “Copernicus: Sentinels Scientific Data Hub,” https://scihub.copernicus.eu, 2016, accessed 2016.08.30.
[379] sentinels.copernicus.eu, “ESA’s Sentinel Online,” https://sentinels.copernicus.eu/web/sentinel/home, 2016, accessed 2016.08.30.
[380] sentinels.space.noa.gr, “Hellenic National Sentinel Data Mirror Site,” http://sentinels.space.noa.gr, 2016, accessed 2016.08.30.
[381] esa.int, “ESA: Observing the Earth,” http://www.esa.int/Our_Activities/Observing_the_Earth, 2016, accessed 2016.08.30.
[382] eoportal.org, “eoPortal: Satellite Missions Database,” https://eoportal.org/web/eoportal/satellite-missions, 2016, accessed 2016.08.30.
[383] openstreetmap.org, “OpenStreetMap,” https://www.openstreetmap.org, 2016, accessed 2016.08.30.
[384] geocommons.com, “GeoCommons Archive,” http://geocommons.com, 2016, accessed 2016.08.30.
[385] en.openei.org, “OpenEI: Open Energy Information,” http://en.openei.org/wiki/Main_Page, 2016, accessed 2016.08.30.
[386] inspire geoportal.ec.europa.eu, “INSPIRE Geo-portal,” http://inspire-geoportal.ec.europa.eu, 2016, accessed 2016.08.30.
[387] inspire.ec.europa.eu, “The E.U. INSPIRE Directive,” http://inspire.ec.europa.eu, 2016, accessed 2016.08.30.
[388] eea.europa.eu, “EEA: Data and Maps,” http://www.eea.europa.eu/data-and-maps, 2016, accessed 2016.08.30.
[389]europa.eu , “EEA: Data Topics,” http://www.eea.europa.eu/themes, 2016, accessed 2016.08.30.
[390] discomap.eea.europa.eu, “European Environment Agency: Discover Map Services,” http://-discomap.eea.europa.eu, 2016, accessed 2016.08.30.
[391] eea.europa.eu, “EEA: Public Map Services,” http://www.eea.europa.eu/code/gis, 2016, accessed 2016.08.30.
[392] europeandataportal.eu, “European Data Portal,” http://www.europeandataportal.eu/en, 2016, accessed 2016.08.30.
[393] data.europa.eu, “E.U. Open Data Portal,” https://data.europa.eu/euodp/en/data, 2016, accessed 2016.08.30.
[394] geodata.gov.gr, “Geodata.gov.gr: Public, Open Data,” http://geodata.gov.gr, 2016, accessed 2016.08.30.
[395] data.gov.gr, “Data.gov.gr,” http://data.gov.gr, 2016, accessed 2016.08.30.
[396] data.gov.ie, “Ireland’s Open Data Portal,” https://data.gov.ie/data, 2016, accessed 2016.08.30.
[397] navigator.eumetsat.int, “EUMETSAT Product Navigator,” http://navigator.eumetsat.int/discovery/Start/Explore/-Quick.do, 2016, accessed 2016.08.30.
[398] eumetsat.int, “European Organization for the Exploitation of Meteorological Satellites,” http://www.eumetsat.int/-website/home/index.html, 2016, accessed 2016.08.30.
[399] iucnrle.org, “IUCN Red List of Ecosystems,” http://iucnrle.org/assessments/, 2016, accessed 2016.08.30.

[400] iucn.org, “International Union for Conservation of Nature,” http://iucn.org/, 2016, accessed 2016.08.30.
[401] earthdata.nasa.gov, “EarthData,” https://earthdata.nasa.gov, 2016, accessed 2016.08.30.
[402] worldview.earthdata.nasa.gov, “EarthData: Worldview,” https://worldview.earthdata.nasa.gov, 2016, accessed 2016.08.30.
[403] eospso.nasa.gov, “NASA’s Earth Observing System,” http://eospso.nasa.gov, 2016, accessed 2016.08.30.
[404] neo.sci.gsfc.nasa.gov, “NEO: NASA Earth Observations,” http://neo.sci.gsfc.nasa.gov, 2016, accessed 2016.08.30.
[405] earthobservatory.nasa.gov, “NASA Earth Observatory,” http://earthobservatory.nasa.gov, 2016, accessed 2016.08.30.
[406] visibleearth.nasa.gov, “NASA Visible Earth,” http://visibleearth.nasa.gov, 2016, accessed 2016.08.30.
[407] espd.gsfc.nasa.gov, “NASA Earth Science Projects Division,” http://espd.gsfc.nasa.gov/index.html, 2016, accessed 2016.08.30.
[408] science.nasa.gov, “NASA Science: Earth Science Data,” http://science.nasa.gov/earth-science/earth-science-data, 2016, accessed 2016.08.30.
[409] earthexplorer.usgs.gov, “USGS: Earth Explorer,” http://earthexplorer.usgs.gov, 2016, accessed 2016.08.30.
[410] arcgis.com, “ESRI Data & Maps,” http://www.arcgis.com/home/-group.html?owner=esri&title=ESRI%20Data%20%26%20Maps&content=all&focus=all, 2016, accessed 2016.08.30.
[411] N. P. Gleditsch, K. Furlong, H. Hegre, B. Lacina, and T. Owen, “Conflicts Over Shared Rivers: Resource Scarcity or Fuzzy Boundaries?” *Political Geography*, vol. 25, no. 4, pp. 361–382, 2006.
[412] marinebio.org, “MarineBio: Marine Conservation Organizations,” http://marinebio.org/oceans/conservation/-organizations, 2016, accessed 2016.08.30.
[413] helcom.fi, “HelCom: Observers,” http://helcom.fi/about-us/observers/international-non-governmental-organisations, 2016, accessed 2016.08.30.
[414] savethehighseas.org, “Deep Sea Conservation Coalition,” http://www.savethehighseas.org, 2016, accessed 2016.08.30.
[415] geodata.myfwc.com, “Florida Fish and Wildlife Research Institute: GIS and Mapping Data Downloads,” http://-geodata.myfwc.com, 2016, accessed 2016.08.30.
[416] creativecommons.org, “Creative Commons (CC),” https://creativecommons.org, 2016, accessed 2016.08.30.
[417] protectedplanet.net, “Protected Planet: Terms & Conditions,” http://www.protectedplanet.net/terms, 2016, accessed 2016.08.30.
[418] unep-wcmc.org, “UNEP-WCMC: Terms and Conditions,” http://www.unep-wcmc.org/terms-and-conditions, 2016, accessed 2016.08.30.
[419] data.unep-wcmc.org, “UNEP-WCMC’s Ocean Data Viewer,” http://data.unep-wcmc.org, 2016, accessed 2016.08.30.
[420] worldweatheronline.com, “World Weather Online: Terms & Conditions,” http://www.worldweatheronline.com/-terms-and-conditions.aspx, 2016, accessed 2016.08.30.
[421] eea.europa.eu, “EEA: Terms & Conditions,” http://www.eea.europa.eu/legal/copyright, 2016, accessed 2016.08.30.
[422] en.wikipedia.org, “Wikipedia: Comparison of free and open-source software licenses,” https://en.wikipedia.org/-wiki/Comparison_of_free_and_open-source_software_licenses, 2016, accessed 2016.08.30.
[423] usgs.gov, “USGS: Copyright permissions,” http://www.usgs.gov/laws/info_policies.html#nonusgs, 2016, accessed 2016.08.30.
[424] reefbase.org, “ReefBase Directory: Use of Contents,” http://www.reefbase.org/useofcontents.aspx, 2016, accessed 2016.08.30.
[425] iobis.org, “OBIS: Ocean Biogeographic Information System - Data Policy,” http://www.iobis.org/node/639, 2016, accessed 2016.08.30.
[426] preview.grid.unep.ch, “Global Risk Data Platform On Natural Events: Permissions,” http://preview.grid.unep.ch/-index.php?preview=about&cat=2&lang=eng, 2016, accessed 2016.08.30.
[427] sentinel.esa.int, “ESA’s Sentinel Online: Terms and Conditions,” https://sentinel.esa.int/web/sentinel/terms-conditions, 2016, accessed 2016.08.30.
[428] openstreetmap.org, “OpenStreetMap: Copyright Terms,” http://www.openstreetmap.org/copyright, 2016, accessed 2016.08.30.
[429] arcgis.com, “ESRI Legal Information,” http://www.esri.com/legal, 2016, accessed 2016.08.30.
[430] J. Lim, Y. Park, J. Lee, D. Seo, and J. Yoo, “An efficient method for processing reverse skyline queries,” in *Mobile Congress (GMC), 2010 Global*. IEEE, 2010, pp. 1–5.
[431] T. Bodea, M. Ferguson, and L. Garrow, “Data set-choice-based revenue management: Data from a major hotel chain,” *Manufacturing & Service Operations Management*, vol. 11, no. 2, pp. 356–361, 2009.
[432] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, “The hadoop distributed file system,” in *Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on*. IEEE, 2010, pp. 1–10.
[433] J. Rowley, “The wisdom hierarchy: representations of the dikw hierarchy,” *Journal of information science*, vol. 33, no. 2, pp. 163–180, 2007.
[434] A. Akdogan, U. Demiryurek, F. Banaei-Kashani, and C. Shahabi, “Voronoi-based geospatial query processing with mapreduce,” in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*. IEEE, 2010, pp. 9–16.
[435] S. Zhang, J. Han, Z. Liu, K. Wang, and S. Feng, “Spatial queries evaluation with mapreduce,” in *Grid and Cooperative Computing, 2009. GCC’09. Eighth International Conference on*. IEEE, 2009, pp. 287–292.

[436] H. Singh and S. Bawa, “A survey of traditional and mapreduce-based spatial query processing approaches,” *SIGMOD Record*, vol. 46, no. 2, 2017.
[437] K. Wang, J. Han, B. Tu, J. Dai, W. Zhou, and X. Song, “Accelerating spatial data processing with mapreduce,” in *Parallel and Distributed Systems (ICPADS), 2010 IEEE 16th International Conference on*. IEEE, 2010, pp. 229–236.
[438] A. Cary, Z. Sun, V. Hristidis, and N. Rishe, “Experiences on processing spatial data with mapreduce,” in *International Conference on Scientific and Statistical Database Management*. Springer, 2009, pp. 302–319.
[439] K.-H. Lee, Y.-J. Lee, H. Choi, Y. D. Chung, and B. Moon, “Parallel data processing with mapreduce: a survey,” *AcM sIGMoD Record*, vol. 40, no. 4, pp. 11–20, 2012.
[440] C. Doulkeridis and K. Nørvåg, “A survey of large-scale analytical query processing in mapreduce,” *The VLDB Journal*, vol. 23, no. 3, pp. 355–380, 2014.
[441] A. Eldawy and M. F. Mokbel, “A demonstration of spatialhadoop: An efficient mapreduce framework for spatial data,” *Proceedings of the VLDB Endowment*, vol. 6, no. 12, pp. 1230–1233, 2013.
[442]Eldawy, A., and mokbel, M. F. , “The ecosystem of spatialhadoop,” *SIGSPATIAL Special*, vol. 6, no. 3, pp. 3–10, 2015.
[443] A. Eldawy, Y. Li, M. F. Mokbel, and R. Janardan, “Cg_hadoop: computational geometry in mapreduce,” in *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2013, pp. 294–303.
[444] A. Eldawy, L. Alarabi, and M. F. Mokbel, “Spatial partitioning techniques in spatialhadoop,” *Proceedings of the VLDB Endowment*, vol. 8, no. 12, pp. 1602–1605, 2015.
[445] A. Eldawy and M. F. Mokbel, “The era of big spatial data: A survey,” *Information and Media Technologies*, vol. 10, no. 2, pp. 305–316, 2015.
[446]Eldawy, A., and mokbel, M. F. , “Pigeon: A spatial mapreduce language,” in *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*. IEEE, 2014, pp. 1242–1245.
[447] A. Eldawy, M. F. Mokbel, and C. Jonathan, “Hadoopviz: A mapreduce framework for extensible visualization of big spatial data,” in *Data Engineering (ICDE), 2016 IEEE 32nd International Conference on*. IEEE, 2016, pp. 601–612.
[448] C. Buchta, “On the average number of maxima in a set of vectors,” *Information Processing Letters*, vol. 33, no. 2, pp. 63–65, 1989.
[449] H. M. V. A. Maximal, *Skyline cardinality for relational processing*. Springer, 2004.
[450] M. Haklay and P. Weber, “Openstreetmap: User-generated street maps,” *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, 2008.
[451] C. Kalyvas, A. Kokkos, and T. Tzouramanis, “A survey of official online sources of high-quality free-of-charge geospatial data for maritime geographic information systems applications,” *Information Systems*, vol. 65, pp. 36–51, 2017.
[452] P. S. Levy and S. Lemeshow, *Sampling of populations: methods and applications*. John Wiley & Sons, 2013.
[453] G. Chandrashekar and F. Sahin, “A survey on feature selection methods,” *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014.
[454] M. B. Kursa, W. R. Rudnicki *et al.*, “Feature selection with the boruta package,” *J Stat Softw*, vol. 36, no. 11, pp. 1–13, 2010.
[455] E. Alpaydin, *Introduction to machine learning*. MIT press, 2009.
[456] L. v. d. Maaten and G. Hinton, “Visualizing data using t-sne,” *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.
[457] P. Baldi, “Autoencoders, unsupervised learning, and deep architectures,” in *Proceedings of ICML workshop on unsupervised and transfer learning*, 2012, pp. 37–49.
[458] C. Zhang and Y. Ma, *Ensemble machine learning: methods and applications*. Springer, 2012.
[459] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, “Do we need hundreds of classifiers to solve real world classification problems?” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3133–3181, 2014.
[460] S. Jeong, K. Chiba, and S. Obayashi, “Data mining for aerodynamic design space,” *Journal of aerospace computing, information, and communication*, vol. 2, no. 11, pp. 452–469, 2005.
[461] K. Doerner, W. J. Gutjahr, R. F. Hartl, C. Strauss, and C. Stummer, “Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection,” *Annals of operations research*, vol. 131, no. 1-4, pp. 79–99, 2004.
[462] P. Godfrey, R. Shipley, and J. Gryz, “Algorithms and analyses for maximal vector computation,” *The VLDB Journal - The International Journal on Very Large Data Bases*, vol. 16, no. 1, pp. 5–28, 2007.
[463] “Kaggle,” https://www.kaggle.com/mustafaali96/weight-height, accessed: 2010-04-22.
[464] C. Kalyvas and T. Tzouramanis, “A survey of skyline query processing,” *arXiv preprint arXiv:1704.01788*, 2017.
[465] C. Kalyvas, T. Tzouramanis, and Y. Manolopoulos, “Processing skyline queries in temporal databases,” in *Proceedings of the Symposium on Applied Computing*, 2017, pp. 893–899.
[466] Y.-C. Chen and C. Lee, “Neural skyline filter for accelerating skyline search algorithms,” *Expert Systems*, vol. 32, no. 1, pp. 108–131, 2015.
[467] T. Kraska, M. Alizadeh, A. Beutel, E. H. Chi, J. Ding, A. Kristo, G. Leclerc, S. Madden, H. Mao, and V. Nathan, “Sagedb: A learned database system,” 2019.
[468] A. Kristo, K. Vaidya, U. Çetintemel, S. Misra, and T. Kraska, “The case for a learned sorting algorithm,” in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 1001–1016.
[469] K. Hose and A. Vlachou, “A survey of skyline processing in highly distributed environments,” *VLDB J.*, vol. 21, no. 3, pp. 359–384, 2012.