Choosing from Skyline sets

K. Touloumis

TU Delft

Choosing from skyline sets

by

K. Touloumis

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday May 27, 2019 at 15:30 PM.

Student number:	4620666	
Project duration:	February 6, 2018 – May 1, 2019	
Thesis committee:	Asst. Prof. dr. Christoph Lofi,	TU Delft, supervisor
	Prof. dr. ir. Geert-Jan Houben,	TU Delft
	Asst. Prof. dr. Casper Bach Poulsen,	TU Delft

This thesis is confidential and cannot be made public until May 15, 2019.

TU Delft

Preface

The need to provide users personalized and tailored access to huge volumes of data has become crucial nowadays due to the exponential growth of information resources. The skyline operator has been proposed to extend traditional database systems and to suggest users optimal objects regarding a specific user preference, that is the skyline set. These are the objects which are not dominated by any other object according to the notion of Pareto dominance. An object dominates another object if it is equal in all dimensions and better in at least one dimension. The skyline set ends up being pretty large because of the effect of what is known as curse of dimensionality. Many reduction algorithms have been proposed to choose interesting objects from the skyline set, but that notion of interestingness is really hard to formalize for different users. A recent study showed that the total number of papers regarding skyline queries is almost 500 up to now, and the initial paper that proposed the skyline operator has been cited more than 2500 times.

This thesis proposes a new way of using skyline algorithms, that is to summarize datasets. The summarization effectiveness of various reduction algorithms will be evaluated in two ways. The first evaluation will compare reduction algorithms on their representativeness. The second will evaluate how the summarization provided by a reduction algorithm affects the completion of a user related task. A framework is proposed for interactive query refinement on multicriteria decision making systems that uses skyline reduction algorithms to summarize query results in order to guide users to their favorable answer.

This master thesis also proposes a new innovative skyline reduction algorithm that first clusters objects, to summarize a dataset, and then applies subspace analysis within each cluster to choose more meaningful objects from each cluster. The design of the algorithm will hopefully outperform already existing reduction algorithms on the grounds of summarizing datasets.

I would like to thank my supervisor, Asst. Prof. Christoph Lofi, for his brilliant idea that initially inspired my thesis, and his excellent co-operation and guidance through my thesis that steered me in the right direction and kept me motivated all the way to the end.

K. Touloumis
Delft, May 2019

Contents

1 Introduction **1**
- 1.1 Context and motivation . . . 1
- 1.2 Thesis contributions . . . 2
- 1.3 Research questions . . . 2
- 1.4 Thesis structure . . . 2

2 Background **5**
- 2.1 Introduction . . . 5
- 2.2 Formalizing Skyline semantics . . . 5
- 2.3 SQL Extension . . . 6
- 2.4 Related problems . . . 7
- 2.5 Skyline Algorithms . . . 7
  - 2.5.1 Block Nested Loop Algorithm . . . 7
  - 2.5.2 Divide and conquer . . . 7
  - 2.5.3 Bitmap . . . 8
  - 2.5.4 Nearest Neighbour algorithm . . . 10
  - 2.5.5 BBS algorithm . . . 10
  - 2.5.6 Constrained skyline queries . . . 11
  - 2.5.7 SFS . . . 11
  - 2.5.8 Dynamic skyline queries . . . 12

3 Reduction Algorithms **15**
- 3.1 Categories of Reduction Algorithms . . . 15
  - 3.1.1 Relaxation of Pareto semantics . . . 15
  - 3.1.2 Summarization approaches . . . 16
  - 3.1.3 Weighting approaches . . . 16
  - 3.1.4 Cooperative approaches . . . 18
- 3.2 Current evaluation methods . . . 19

4 Proposed way of using skyline sets **21**
- 4.1 Introduction . . . 21
- 4.2 Current applications . . . 21
- 4.3 Proposed framework . . . 22
- 4.4 Evaluating a summarization . . . 23
- 4.5 Intrinsic evaluation . . . 23
- 4.6 Extrinsic evaluation . . . 23

5 Clustering Based Skyline frequency **25**
- 5.1 Introduction . . . 25
- 5.2 Motivation . . . 25
- 5.3 Clustering . . . 25
- 5.4 Clustering Based Frequency . . . 25
- 5.5 Evaluation . . . 26
  - 5.5.1 Effect of dimensionality . . . 26
  - 5.5.2 Summarization . . . 27
  - 5.5.3 Discussion . . . 27

6 Study setup **31**
6.1 Case Study setup . . . 31
6.1.1 Objectives . . . 31
6.1.2 Background . . . 31
6.1.3 System's Overview . . . 32
6.1.4 Algorithms for summarization . . . 32
6.1.5 Dataset for case study . . . 32
6.1.6 Participants . . . 33
6.1.7 Task . . . 33
6.1.8 Independent Variables . . . 33
6.1.9 Dependent Variables . . . 33
6.1.10 Blocking factors, noise factors and co-variances . . . 33
6.1.11 Data sources and Instrumentation . . . 33
6.1.12 Procedure . . . 34
6.1.13 Units of measurements . . . 34
6.1.14 Assumptions . . . 34
6.2 Dataset preparation . . . 35
6.2.1 Data cleaning . . . 35
6.2.2 Data binning . . . 35

7 Findings **39**
7.1 Findings on representativeness . . . 39
7.2 Findings of case study . . . 40
7.3 Conclusions . . . 43

8 Conclusion and future work **45**

A List of tables **47**

B List of figures **49**

C Questionnaire **51**
C.1 Personal details . . . 51
C.2 Algorithm evaluation . . . 51
C.3 Self assessment method . . . 52
C.4 Task description . . . 52

Bibliography **53**

1 Introduction

1.1. Context and motivation

As we march towards this new era of "Big Data" [36], information sources, including recommendation systems, social networks, transactional systems, mobile devices have seen an exponential growth. Businesses' IT departments try to come up with new innovative systems in order to leverage immense amounts of information. Nowadays with all this flooding information it is imperative to provide users personalized and monitored access to huge volumes of data.

Skyline queries [3] have been proposed to fill the gap between traditional and multimedia database systems [15]. In traditional database systems the answer to a query is an unordered set of records, whereas in multimedia database systems the answer is a graded set of objects. Multimedia database systems employ preference functions, that require explicit user elicitation, in order to establish ranking between objects. On the other hand, skyline queries can be computed in a more generic manner based on a personal user preference.

Skyline queries are a popular way of obtaining the dominant objects from a database. These are the objects which are not dominated by any other object according the notion of Pareto dominance [17], an object dominates another if it is better in at least one attribute and equal in all others. The Pareto dominance can be used as a personalized filter to rule out objects that are dominated for any user preference resulting in the skyline set of a query.

The Pareto dominance was initially designed to handle only numerical preferences [3, 26]. However, throughout the years it was extended to handle categorical preferences as well, modelled as weak order preferences [5]. What is really interesting is that most of these preferences do not require any specific user elicitation as they can be extracted in a more generic manner based on user profiles. For example between two products with the same characteristics, it can be stated without any loss of generality that the product with lower price will be more preferable. That makes skyline sets fair and easy to be computed.

So far the skyline operator has been used to extend traditional databases trying to extract interesting objects for users. Skyline algorithms have been used in multiple criteria decision making and decision support systems [24]. For example, skyline queries have been used in location based systems [20] trying to find the on route shortest way to a destination. Also, skyline algorithms have been used by travel agencies to find cheap hotels within a particular distance [3]. Reverse skyline queries [14] are used in market research in order to find the best location for a new branch store to open or to decide which product would be more appealing to customers. Lately, skyline algorithms have attracted particular interest in bioinformatics [23], representing DNA sequences and performing efficient pattern searching.

One of the major shortcomings of skyline computation is that skyline sets usually end up being pretty large because of the effect of what is known as curse of dimensionality [8]. As the number of dimensions increases it is more likely that more and more objects will become incomparable and will end up in the skyline set. In [3] a case study proved that for 10 attributes the skyline set can be almost 30% of the initial database making skyline sets unmanageable to most users and inapplicable to real time scenarios.

In order to choose interesting objects from skyline sets so as to increase their manageability, many reduction algorithms [31] have been proposed. However, there has not been a standard definition of "interestingness" up to now. It can be seen in literature that each author interprets interestingness in a different way,

which is usually a mathematical principle. For example in [7] the authors claim interesting a point that has a high skyline frequency. Similarly, in [6] authors claim meaningful a point that is not k-dominated. Again in exactly the same way interestingness varies among users as well and is hard to formalize.

1.2. Thesis contributions

Based on the material on reduction algorithms presented in [31], the following new contributions will be made:

- One of the very first contributions of this master thesis is to propose a new way of using reduced skyline sets, that is to summarize query results. Instead of using skyline reduction algorithms as an extension of traditional databases, use reduction algorithms along with traditional querying techniques to summarize query results. Then, users can refine their query, which of course they will do much more accurately once they already have a good overview. Specifically, a framework will be suggested for interactive and progressive query refinement, that will allow users to refine their queries by first giving them an overview of their possible choices right after performing a query.
- On the grounds of summarizing query results it would be highly reasonable to compare all reduction algorithms on their summarization effectiveness. However, up to now there has not been proposed a metric for that purpose. Some metrics from the fields of statistical analysis and data mining will be examined, then a new user related metric will be proposed, and a case study will be conducted to measure the summarization effectiveness of various reduction algorithms on that new metric. The results of the case study combined with the results of the metrics already stated in literature will be discussed, reduction algorithms will be compared against each other, and some very interesting conclusions will be drawn regarding their summarization effectiveness.
- In the context of this master thesis a new reduction algorithm was developed, namely Clustering Based Frequency, that in order to choose more meaningful objects from the skyline set, first clusters objects into neighborhoods of interestingness by applying a clustering algorithm, and then finds the most dominant objects within each cluster by performing subspace analysis. Two variations of the algorithm will be presented. The design of this reduction algorithm will hopefully outperform already implemented reduction algorithms on the grounds of summarizing query results.

1.3. Research questions

The research questions that will be addressed throughout this thesis are the following:

- Can skyline reduction algorithms be used to summarize query results?
- Are metrics in literature adequate to evaluate the summarization provided by a skyline reduction algorithm?
- How to evaluate the summarization provided by a reduction algorithm?
- How do the reduction algorithms of k-Dominant skylines, Skyline Frequency, Random Reduction, Clustering Based Frequency, Distance Based Representatives perform on that new notion of summarization?
- Does Clustering Based Frequency provide a more meaningful summarization?

1.4. Thesis structure

The remainder of this thesis is organized as follows:

Chapter 2, deals with the problem of formalizing skyline semantics, the skyline operator and its properties are presented, some related problems to skyline queries are mentioned, and a small literature survey is performed on skyline algorithms.

Chapter 3 briefly mentions the most prominent reduction algorithms. Also, current methods for evaluating reduced skyline sets are mentioned.

Chapter 4 firstly proposes a new way of using reduced skyline sets, that is to summarize datasets. Then the problem of evaluating the summarization provided by a reduction algorithm is addressed. Firstly, two metrics

are borrowed from literature, namely sample representativeness from statistical analysis and automatic summarization from data mining, then the notion of summarization is redefined in a more user related manner, and a user experiment is designed that will measure the summarization effectiveness of the five reduction algorithms tested for the shake of this master thesis.

Chapter 5 presents the Clustering Based Frequency algorithm, two variations of the algorithm will be discussed and compared against each other on the performance and the dimensionality effect. Finally, the algorithm will be compared against already existing summarization approaches to prove that it gives a more meaningful summarization of the full skyline set.

Chapter 6 describes the experimental setup. Firstly, the controlled user study that will be conducted is analytically explained. Also, the dataset that will be used is described and all the steps of preprocessing it. These are the data preparation and data cleaning, then the binning of data is described and dominance relations are explained for both numerical and categorical attribute domains.

Chapter 7 presents and discusses the results of the metrics stated in literature along with the results of the user experiment.

Chapter 8 concludes the thesis with some remarks and future outlooks.

2
Background

2.1. Introduction

This chapter firstly introduces the skyline operator, its semantics and properties from already existing work. Then, some related problems to the skyline computation are briefly mentioned. Finally, some algorithms are presented for computing the skyline set.

2.2. Formalizing Skyline semantics

Let $A = \{A_1, A_2, \ldots, A_d\}$ be a finite set of d attributes where each A_i is associated with an attribute domain D_{A_i}, and a dataset $Ds = \{p_1, p_2, \ldots, p_d\}$ where each p_i is a $d-$dimensional data point on A. The value of data point p_i on attribute A_j will be denoted by $p_i.v_j$. A preference can be formalized in the following way similarly as in [31]:

- A preference $P_i = (a, b)$ defines a strict partial order over attribute domain D_{A_i}. Such a preference declares that value a is preferred over value b. This will be denoted as $a >_i b$, a dominates b on attribute domain D_{A_i} regarding preference P_i.
- Similarly, if two values a, b are equivalent on attribute domain D_{A_i}, the equivalence relation will be defined as $a \approx_i b$,
- If value a is preferred over, or equivalent to value b on attribute domain D_{A_i}, that is denoted as $a \geq_i b$.

Some example partial order preferences will be displayed in figure 2.1 on the domain of cars. Three partial order preferences are displayed for the attributes of price, mileage and body-style from left to right.

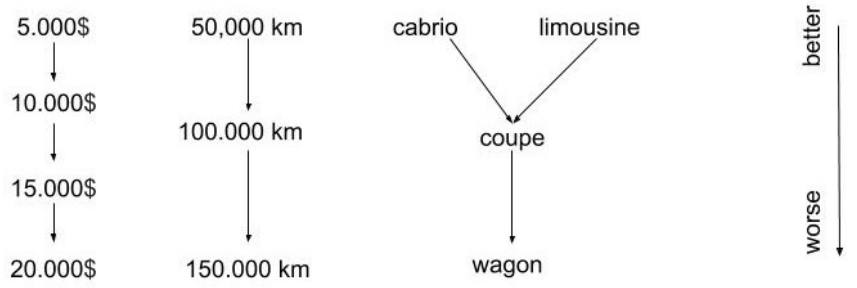


Figure 2.1: Partial order preferences on the attributes of price, mileage, body-style

With having defined the basics on preferences, the next step is to define the dominance relation between two data points.

Dominance relation: Given two dataset points p_1 and p_2, p_1 dominates p_2 when p_1 is better in at least one attribute than p_2 and better or equal in all others.

$$p_1 > p_2 \iff \exists i \in [1,d] : p_1.v_i >_i p_2.v_i \wedge \forall j \in [1,d] - i : p_1.v_j \geq_j p_2.v_j$$

For the dominance relation the transitivity also holds.

Transitivity: Given three points $a, b, c \in Ds$, if $a > b$ (a dominates b) and $b > c$ (b dominates c) then $a > c$ (a dominates c)

Based on the previous partial order preferences we can draw the conclusion that a car with attribute values (10.000$, 50.000km, cabrio) is more preferable over a car with values (15.000$, 100.000km, cabrio). However, it is not always the case that such a dominance relation can be established, and that can happen due to many reasons. Firstly, two objects might have antagonistic values for different attributes e.x when comparing two cars (15.000$, 50.000km, cabrio) and (10.000$, 100.000km, cabrio). We can see the first car dominates the second car on the mileage, but the second car dominates the first one on the price, so no dominance relation can be established on these two cars. In that case these two objects are incomparable. Secondly, it might be the case that there is no preference relation defined between two attribute values. For example, on the preferences showed before it is not clear what is the relation between a cabrio bodystyle and a limousine. Finally, the user maybe is indifferent regarding some attribute values like "A cabrio bodystyle is as preferable as a limousine bodystyle".

Incomparability: Given two distinct dataset points $a, b \in Ds$, if neither $a > b$(a dominates b) nor $b > a$(b dominates a) then the two points are considered incomparable.

Now everything is ready to define the skyline set.

Skyline set The skyline set contains all the data set points that are not dominated by any other point.

$$SKY = \{p \in Ds \mid \neg \exists\, o \in Ds : o > p\}$$

Skyline queries incorporate user preferences to extract interesting objects, and can be used as a personal filter to rule out objects that are not particularly interesting for any user preference. As mentioned in [24] one nice property of skyline sets is that any set of evaluation criteria that arise from user's preferences can be modeled as a monotone scoring function like the L_1 or L_2 norm. Every point that will be in the skyline set will maximize or minimize that scoring function. This means that regardless how a user weights his personal preferences towards attribute values, the most favourable answer to every query will be found in the skyline set. Another property of the skyline set is that all points in it are incomparable against each other. Although the points in the the skyline set are the most dominant points, there will be no skyline point that will best satisfy a user preference. Instead, there will be many answers in the skyline set that will be close to the user's satisfaction.

2.3. SQL Extension

The skyline operator was proposed in [3] where the authors suggested to extend the SQL *SELECT* statement by an optional SKYLINE OF clause. The skyline clause shall be placed after the join and group by clauses and before the sort clause. In the skyline clause the user should determine the attributes for the skyline computation and the criteria for them, which can be one of the following directives, *MIN*, *MAX* or *DIFF*. The syntax proposed is the following:

$$SELECT < attributes > FROM < relations > WHERE < conditions > GROUPBY < attributes >$$
$$HAVING < conditions > SKYLINE\ OF\ d1[MIN|MAX|DIFF], \ldots, dn[MIN|MAX|DIFF]$$

After the SKYLINE OF clause the attributes must be placed one after another along with their directives. The directives MIN, MAX, DIFF specify that the corresponding attribute must be minimized, maximized, be different(distinct) from all others. It is worth mentioning that a one dimensional skyline query can be constructed in a straightforward manner by just computing the min and max of the attribute the user is interested in.

2.4. Related problems

This section presents some problems similar to the skyline computation as discussed in [24].

The first ever problem to deal with retrieving the optimal points was the vector maximization problem [27]. The vector maximization problem is about maximizing n d-dimensional vectors on a d-dimensional vector-valued criterion function. The efficient computation of vector maximization has received great attention and must be definitely considered as the forefather of skyline queries.

Convex hull [35] algorithms return an area of interesting points bounded by a polygon defined by the minimum and maximum distances. It can be noticed that convex hull algorithms differ from the skyline algorithms as they return an area of interesting points and not a single line with individual points.

A Top-k query [21] returns the best k objects evaluated on a scoring(preference) function, these are the k objects that score the k higher or lower results for that function. Obviously the best k points retrieved are not necessarily part of the skyline set. According to [46] skyline queries are closely connected to top-1 queries, because a top-1 query returns the object with the best score on the scoring function. If the scoring function is monotone, then the top-1 result will be in the skyline set. Conversely, every skyline point will be a top-1 result for at least one preference function. Usually that scoring function is an aggregated function of partial scores. Top-k queries are of great importance to meta search engines, where results need to be efficiently aggregated and ranked from different search engines.

Nearest Neighbor algorithms [12] are the simplest non parametric classification models that assign to a new object which belongs in the test set, the category label of its nearest sample in the training set. The main difference of nearest neighbor techniques and skyline queries are that nearest neighbor techniques do not in any way engage dominance relations, instead they depend on distance based metrics.

2.5. Skyline Algorithms

In this section the most important algorithms for computing the skyline sets will be summarized. Skyline algorithms can be divided in two categories, non-index based methods, that do not exploit any special data structure for computing the skyline set, and index based methods, that use a spatial data structure like multidimensional R-trees to compute the skyline set much more efficiently.

2.5.1. Block Nested Loop Algorithm

The block nested loop(BNL) algorithm [3] is the simplest way that the skyline set can be computed. The basic idea of this algorithm is to compare every object in the database with every other object. Obviously the BNL algorithm belongs to the family of naive algorithms, that works in a double for loop. A window is maintained that stores the incomparable objects. Obviously, during the first iteration the window is empty, so it must be initialized with the first object of the database. During each iteration of the algorithm an object p is fetched from the database and it is compared against the window. Three cases can occur:

- If p is dominated by any object in the window, then p will be eliminated and will not be considered for future iterations.
- If p dominates one or more objects in the window, firstly the dominated objects will be removed from the window and will not be considered for future iterations, and secondly p will be inserted into the window.
- If p is incomparable with all objects in the window then it will be inserted into the window.

At the end of the algorithm the window will be returned as the output which will of course contain all the dominant objects. The worst case complexity of the algorithm is $O(N^2)$ where N is the number of objects in the database. The logic of the algorithm is also depicted in figure 2.2.

2.5.2. Divide and conquer

The Divide and Conquer(DC) algorithm proposed in [3] recursively divides the space into smaller partitions by computing the median for one dimension at a time. The full skyline is computed by merging the skyline sets of the partitions. The overall complexity of the algorithm is $O(N(logN)^{d-2}) + O(NlogN)$, where d is the number of dimensions and N the total number of objects.

Figures 2.3 and 2.4 display a two-way and three-way partitioning respectively. Taking first the two-way partitioning, it can easily be seen that all objects in partition $S_{1,1}$ will be in the skyline set. On the contrary

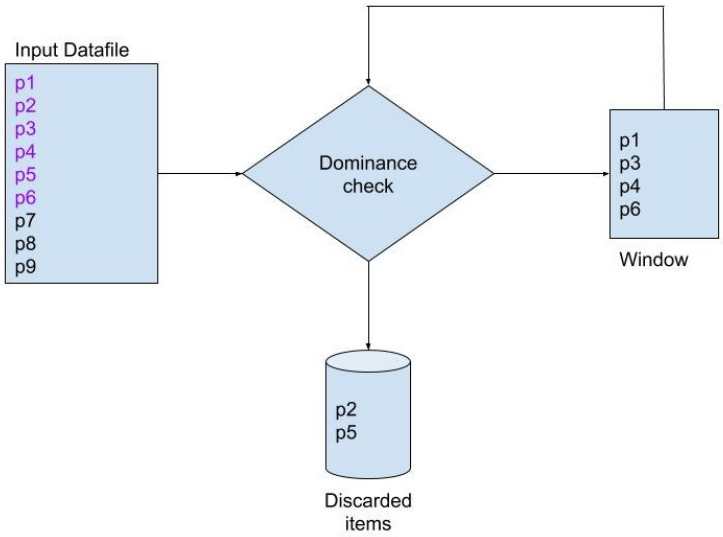


Figure 2.2: The BNL algorithm for an input data file of 9 objects. The next object to be processed is $p7$

all objects in $S_{2,2}$ can be discarded immediately as they are dominated by any point in $S_{1,1}$. The skyline sets for partitions $S_{1,2}$ and $S_{2,1}$ must be computed, merged and checked for dominance relation's with skyline points in $S_{1,1}$. Analogously, for the three way partitioning points in partitions $S_{2,2}$ and $S_{2,3}$ can be discarded immediately as they are dominated by any point in $S_{1,1}$ and skyline sets must be computed for partitions $S_{1,2}$, $S_{1,2}$, $S_{2,1}$, checked and merged with skyline points in $S_{1,1}$

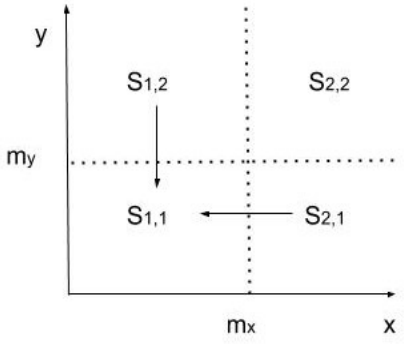


Figure 2.3: 2-way partitioning

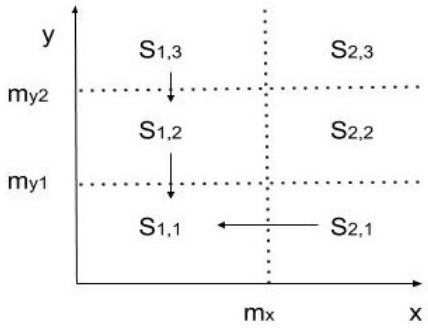


Figure 2.4: 3-way partitioning

2.5.3. Bitmap

In [45] a progressive algorithm for computing the skyline set was proposed that maps all database objects onto a bitmap structure to exploit bitwise operations. The algorithm doesn't need to scan the entire database, instead the preprocessing makes it possible to stop the algorithm at any time and return the skyline set up to that point.

To encode the database objects the following scheme is used. Each object is mapped onto a m-bit vector, where m is the sum of total numbers of distinct values from each of d dimensions, so $m = \sum_{i=1}^{d} k_i$. Suppose there are k_i distinct values on dimension i then the j_ith smallest value is represented by k_i bits where the leftmost $k_i - j_i + 1$ bits are 1 and the remaining bits are 0. To judge whether a single point belongs to the skyline set, a bitwise AND operation is applied on two bit string V_x and V_y, where V_x and V_y are formed by juxtaposing the rightmost corresponding bits from the encoding of every point. If the resulting bitmap has only one 1 then the point belongs to the skylines set, otherwise it doesn't.

An example will be displayed on a data file of 11 objects, representing rooms for rent in Delft as depicted in 2.5. The code for creating and plotting the datafile can be found in [47]. The x-axis shows the price in euros per month and the y-axis the distance from the city centre in meters. The blue line connects the skyline points. It is obvious that a room is more preferable when its both close to the city centre, and costs less per month.

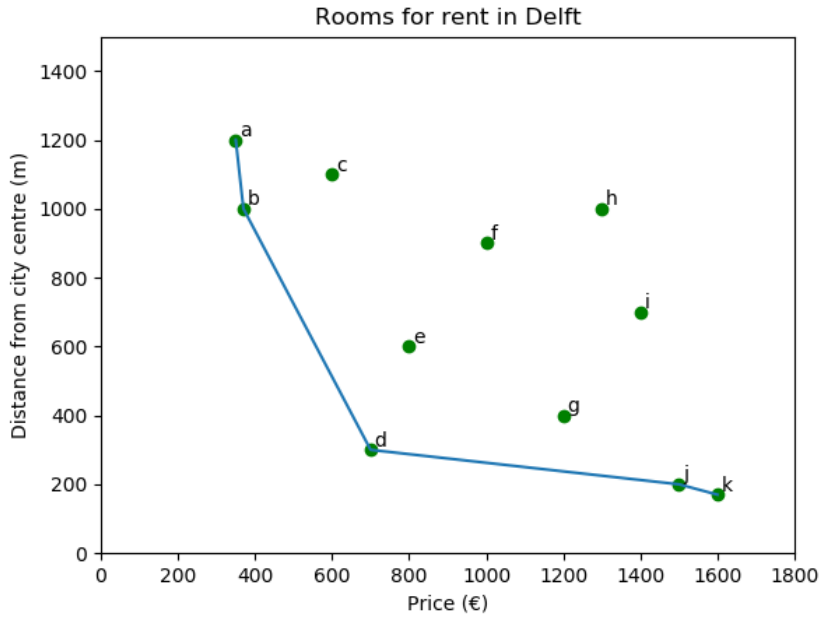


Figure 2.5: The rooms data file along with its skyline set(the points connected with blue line)

Table 2.1 below shows the bitmap encoding for each point.

Point	**Bitmap representation**
a(350, 1200)	(11111111111, 1000000000)
b(370, 1000)	(11111111110, 1110000000)
c(600, 1100)	(11111111100, 1100000000)
d(700, 300)	(11111111000, 1111111100)
e(800, 600)	(11111110000, 1111110000)
f(1000, 900)	(11111100000, 1111000000)
g(1200, 400)	(11111000000, 1111111000)
h(1300, 1000)	(11110000000, 1110000000)
i(1400, 700)	(11100000000, 1111100000)
j(1500, 200)	(11000000000, 1111111110)
k(1600, 170)	(10000000000, 1111111111)

Table 2.1: Bitmap encodings of example data file

It can be checked for point $b(370, 1000)$ that we must take the second rightmost bits from the x encodings because 370 is the second smallest value among x's, that makes $V_x = 11000000000$. As for the y-encodings 1000 is the second largest value, so we must take the eighth rightmost bits, that gives $V_y = 01011111111$.

$$V_x \; AND \; V_y = 01000000000$$

The result has only one 1, so point b belongs to the skyline set. For point e we must take the fifth rightmost bits from the x-encodings, so $V_x = 11111000000$, and the fifth rightmost values from the y-encodings, $V_y = 00011010011$.

$$V_x \; AND \; V_y = 0001100000$$

The result has two 1s, so point e does not belong to the skyline set.

2.5.4. Nearest Neighbour algorithm

Index based methods rely on spatial data structures so as to perform the skyline computation in a more dynamic manner and to massively eliminate points in order to reduce redundant checks. The very first of them was the Nearest Neighbor(NN) algorithm[26], that uses a multidimensional R tree. The algorithm recursively applies nearest neighbor search to the region spanned by the origin to find skyline points. The metric used for the nearest neighbor search has to be a monotone distance function e.x the Euclidean distance. Every region that must be checked is entered into the *todo* list and will be checked in the next iteration. Every region that is already dominated can be immediately discarded. One disadvantage of the NN algorithm is that it has an increased I/O complexity. Also the size of the *todo* list may exceed the dataset size.

An example will be explained on the data file with rooms for rent in Delft. The room with the minimum euclidean distance from the origin is point d, which is added to the skyline set. Room d divides the space in 3 partitions $R1$, $R2$, $R3$. Region $R3$ can immediately be discarded because every point in it is dominated by point d. $R1$ and $R2$ are inserted to the *todo* list. The closest point to $R1$'s origin is point b which is added to the skyline set. Point b divides $R1$ in three partitions, $R4$, $R5$ and $R6$. Partition $R6$ is empty and partition $R5$ is discarded because all points in it are dominated by point b. $R4$ is inserted into the *todo* list which now has $R2$, $R4$ and the skyline set is d, b. The closest point to $R2$'s origin is j and it is added to the skyline set. Point j divides $R2$ in partitions $R7$, $R8$, $R9$. $R7$ and $R9$ are empty and $R8$ is added to the *todo* list. $R4$ has only one item, point a, which is added to the skyline set and $R8$ has only one item point k which is also added to the skyline set. At the end the *todo* list is empty and the skyline set contains rooms a, b, d, j, k. Figures 2.6 and 2.7 illustrate and example of partitioning the data space.

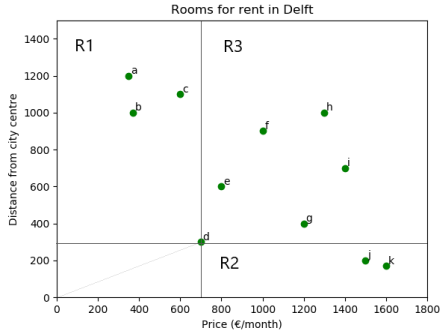


Figure 2.6: Partitioning space with skyline point d

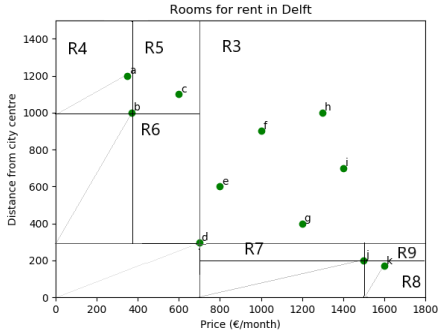


Figure 2.7: Partitioning space with skyline points d, b, j

2.5.5. BBS algorithm

The branch and bound skyline algorithm [40] is a progressive algorithm that works similarly to the NN but it is I/O efficient, the R-tree nodes that contain skyline points are accessed only once. An intermediate entry of the R-tree represents a minimum bounding rectangle(MBR) of a node, and a leaf node represents a data point. The metric used this time is the L_1 norm. The *mindist* of a leaf node equals the sum of its coordinates and the *mindist* of an intermediate node equals the *mindist* of its lower left point. The algorithm recursively traverses the tree starting from the root node and at each iteration the node with the minimum *mindist* is expanded by inserting its children to the heap.

An example will be displayed on the rooms data file indexed by the R-tree depicted in 2.9 with corresponding minimum bounded rectangles as in figure 2.8. At the beginning the heap is initialized with R1 and R2 and their corresponding minimum distances. $H = \{(R1, 950), (R2, 750)\}$. R2 has the smaller *mindist* so it is expanded. $H = \{(R5, 930), (R6, 1500), (R1, 950)\}$. Now R5 is expanded and the heap becomes $H = \{(e, 1400), (d, 1000), (g, 1600), (R6, 1500), (R1, 950)\}$. Point d has the minimum *mindist* so it is inserted to the skyline set, $S = \{d\}$. Both e and g are discarded because they are dominated by d. $H = \{(R6, 1500), (R1, 950)\}$. Now R1 is expanded $H = \{(R3, 1330), (R4, 1650), (R6, 1500)\}$. Next R3 is expanded and the heap becomes $H = \{(a, 1550), (b, 1370), (c, 1700), (R4, 1650), (R6, 1500)\}$. Point b has the minimum *mindist* and is inserted to the skyline set, $S = \{d, b\}$. Point c and entry R4 are discarded because they are dominated by b, and a is inserted to the skyline set because it is incomparable with both b and d. $H = \{(R6, 1500)\}$ and $S = \{a, b, d\}$. R6 is expanded and the heap becomes $H = \{(j, 1700), (k, 1770)\}$ and $S = \{a, b, d\}$. Both points j and k are incomparable with the skyline set so they both are appended to that. Finally, the skyline set is $S = \{a, b, d, j, k\}$ and the heap is

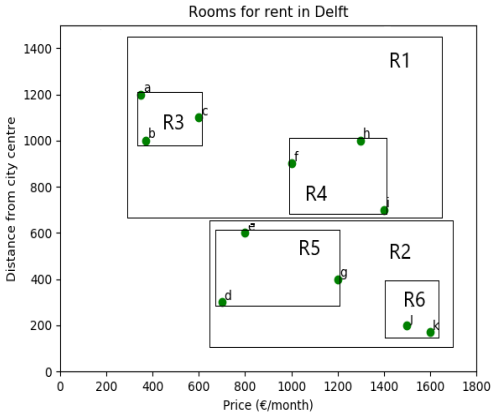


Figure 2.8: Minimum bounded rectangles

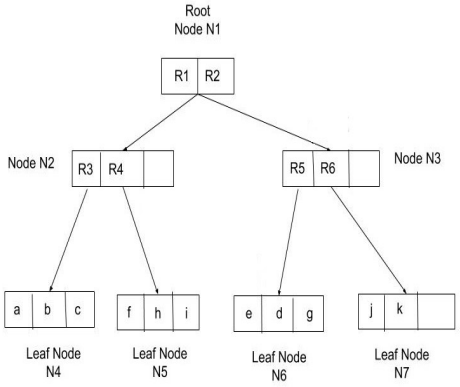


Figure 2.9: R-tree on the data file

empty.

2.5.6. Constrained skyline queries

Constrained skyline queries [13] help users explore a particular subset of data that might be more interesting for them by restricting the entire data space based on a set of constraints. Each constraint is formulated as a range along each dimension. A constrained skyline query returns the most interesting objects defined by these constraints. Constraints on skyline queries can be expressed in two ways. There are constrained skyline queries where the skyline set is computed after the dataset is restricted by the constraints, and skyline queries with constraints, where the skyline set is computed on the entire dataset and then gets restricted by the constraints returning the most interesting skyline points to the user.

On the previous example on the data file with rooms for rent in Delft a user might be interested in finding rooms that do cost between 300 and 1300 euros per month within the distance of 200 and 1000 meters away from the city centre. The two SQL queries are given, the first is for constrained skyline queries, and the second for skyline queries with constraints and figures 2.10, 2.11 depict the difference between them.

Constrained skyline query:

SELECT * FROM Rooms WHERE ((distance≥ 200 AND distance ≤ 1000) AND (price ≥ 300 AND price ≤ 1300)) SKYLINE OF price MIN, distance MIN

Skyline query with constraints:

SELECT * FROM (SELECT FROM Rooms SKYLINE OF price MIN, distance MIN) skyline_set WHERE ((distance≥ 200 AND distance ≤ 1000) AND (price ≥ 300 AND price ≤ 1300))

2.5.7. SFS

The Sort Filter Skyline algorithm, proposed in [11], extends the BNL algorithm. The SFS algorithm demonstrates a progressive behaviour by presorting data points based on a monotone preference function. One example of a monotone preference function is the sum of normalized coordinates on all dimensions of a point, named as entropy. Presorting in ascending order makes sure that a point a dominating point b will be visited before point b, and the number of comparisons will be reduced to the minimum. That ensures the progressive behaviour of the algorithm. After the presorting phase the execution of the algorithm is the same as the BNL algorithm.

Empirical studies have shown that the most effective entropy function of a data point p is the $E_d(p) = \sum_{i=1}^{d} ln(p.v_i + 1)$ where $p.v_i$ is the normalized value of a data point p on dimension i in the interval $(0,1)$ non-inclusive. The smaller the entropy of a data point is the more dominant the point will probably be.

The complexity of the algorithm is $O(dN^2)$ in the worst case and $O(dN + NlogN)$ in the best case, where d is the number of dimensions and N is the number of points in the dataset. Although the SFS algorithm reduces radically the number of comparisons, still the entire dataset has to be scanned to compute the full skyline set. Also the algorithm cannot adapt to different user preferences.

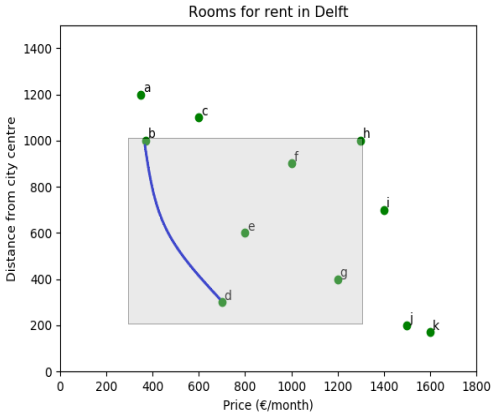


Figure 2.10: Constrained skyline query, $S = b, d$

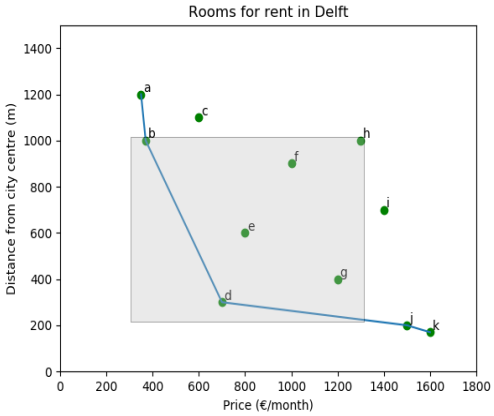


Figure 2.11: Skyline query with constraints, $S = b, d$

An example will be demonstrated on the data file with rooms for rent in Delft. Since all values must be normalized in the interval (0, 1) non inclusive, all values were divided by a large number, 2000. Table 2.2 shows the normalized values of the two attributes for every room in the data file and the corresponding computed entropy. With the table now sorted in ascending order the algorithm proceeds in a similar way with the BNL algorithm. Each point is fetched, d is the first point and it is inserted in the window, b is incomparable with d and is inserted to the window as well. Point e is dominated by point d so it is discarded, point a is incomparable with all objects in the window and is inserted in the window. Point g is dominated by point d so it is discarded. Points j and k are incomparable with the window objects and are inserted in the window. All the other points are dominated and discarded.

Room	Normalized Price	Normalized Distance	Entropy
d	0.35	0.15	0.440
b	0.185	0.5	0.575
e	0.4	0.3	0.599
a	0.175	0.6	0.631
g	0.6	0.2	0.652
j	0.75	0.1	0.655
k	0.8	0.085	0.669
c	0.3	0.55	0.701
f	0.5	0.45	0.777
i	0.7	0.35	0.831
h	0.65	0.5	0.906

Table 2.2: Presorting based on entropy computation

2.5.8. Dynamic skyline queries

Dynamic skyline queries [10] map all points on a new data space by finding their dynamic coordinates based on a set of distance functions and a reference point. Dynamic skyline queries define a new $d'-$dimensional data space from the original d-dimensional data space. The new data space is called dynamic space.

In order to achieve that transformation m distance functions (with $m \leq d$) must be defined. Every function takes as input the original coordinates of a point and the reference point q and returns the point's dynamic coordinates. To make things simple it is assumed that $d = d'$, the dynamic data space has the same dimensionality as the original data space, and every distance function on dimension i is defined as the absolute distance between the original coordinates and the coordinates of the reference point, $f_i(p, q) = \left|q_i.v_i - p_i.v_i\right|$. Now the dominance relation must be redefined for dynamic skyline queries.

Dynamic dominance: Given two data set points p_1, p_2 and a reference point q, p_1 dynamically dominates p_2

Point	Original Coordinates		Dynamic Coordinates	
	price	distance	price	distance
a	350	1200	1050	1200
b	370	1000	1030	1000
c	600	1100	800	1100
d	700	300	700	1100
e	800	600	800	800
f	1000	900	1000	900
g	1200	400	1200	1000
h	1300	1000	1300	1000
i	1400	700	1400	700
j	1500	200	1500	1200
k	1600	170	1600	1230
q	700	700	700	700

Table 2.3: Original and dynamic coordinates

$$p_1 >_q p_2 \iff \exists i \in [1,d] : \left|p_1.v_i - q.v_i\right| > \left|p_2.v_i - q.v_i\right| \wedge \forall j \in [1,d] - i : \left|p_1.v_j - q.v_j\right| \geq \left|p_2.v_j - q.v_j\right|$$

Dynamic Skyline: The dynamic skyline set contains all points that are not dynamically dominated by any other point.

$$DSKY = \{p \in Ds \mid \neg\exists\, o \in Ds : o >_q p\}$$

An example of dynamic skyline queries will be illustrated on the rooms data file with reference point $q = (700, 700)$. Table 2.3 shows the points' transformation, and figures 2.12, 2.13 the dynamic data space and dynamic skyline set respectively.

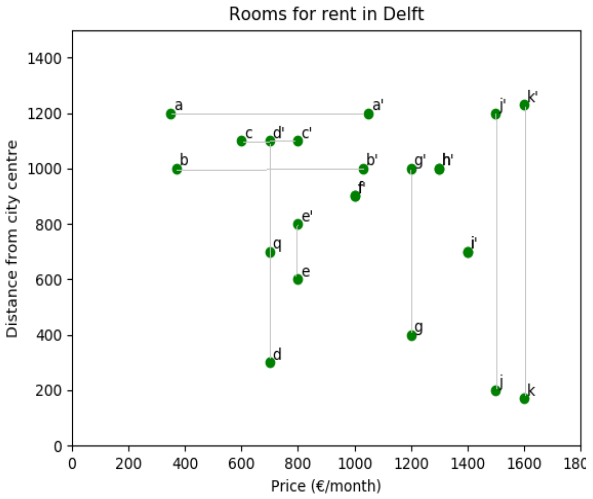


Figure 2.12: Dynamic data space

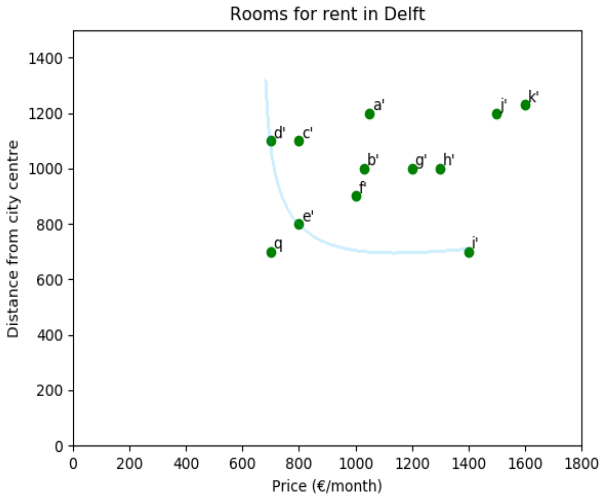


Figure 2.13: Dynamic skyline set

Dynamic skyline queries are very effective when it comes down to detecting interesting objects from a user's perspective. The reference point $q = (700, 700)$ led to the skyline of d, e, i. That can simply be translated into "A user that was interested in room q would probably be very much interested in rooms d, e and i as well".

3

Reduction Algorithms

This chapter deals with reduction algorithms. The first section outlines the basic categories of reduction algorithms while the second section critically assess the way reduction algorithms are evaluated in literature.

3.1. Categories of Reduction Algorithms

According to [31] reduction algorithms can be divided into 4 categories:

3.1.1. Relaxation of Pareto semantics

According to the notion of Pareto dominance if two objects are incomparable regarding a single attribute then they become incomparable and maybe they will both end up in the skyline set. It can be seen that the full Pareto dominance is a very strict notion of dominance. Algorithms belonging to this category try to relax the notion of Pareto dominance, leading to more dominance relations and as a result to smaller skyline sets. The most prominent example of this family of reduction algorithms are k-dominant skylines [6]. k-dominant skylines weaken the notion of Pareto dominance by introducing the k-dominance relation. An object k-dominates another object if it is better or equal in at least k dimensions where $k < d$ and strictly better in at least one of these k dimensions. The main idea behind k-dominant skylines is that an object displaying many good attribute values will dominate another one displaying a few good attribute values. If one object k+1-dominates another object then it will k-dominate it as well. When k equals the total number of attributes, $k = d$, the k-dominant skyline becomes the full skyline set. As the parameter k decreases the size of the skyline set decreases as well. Figure 3.1 shows the hierarchy of k-dominant skylines.

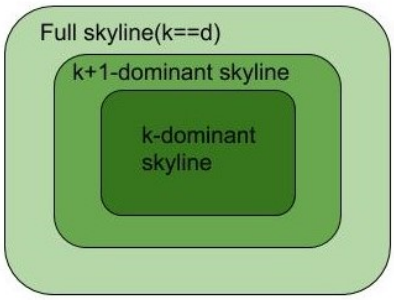


Figure 3.1: Full skyline set and k-dominant skyline sets

Algorithms in this category, so called weak order skylines, can be designed in a straightforward manner from classical skyline algorithms by just modifying the notion of dominance. Unfortunately, research has shown that these algorithms significantly decrease the size of skyline sets and probably some very interesting

skyline objects will be discarded as well. The summarization effectiveness of this family of reduction algorithms is doubted.

3.1.2. Summarization approaches

The purpose of summarization approaches is to return a summarization of the skyline set, that is a set of objects which still maintains the diversity of attributes of the full skyline set but is much smaller in size. The main idea behind summarization techniques is to give users a taste of different neighbourhoods of interestingness in the skyline set

An example of this family is Approximate Dominating Representatives(ADR)[25]. The basic idea is to return the best sample of the skyline set in terms of size and accuracy. The way to achieve that is to return a set of objects $adr = \{a_1, ..., a_k\}$ such that for any user specified value ϵ, any other other object $o \notin adr$ holds that there is an $i \leq k$ such that the vector $a_i \times (1+\epsilon)$ dominates object o. The idea is illustrated in figure 3.2. All summarization approaches are a post processing step of the skyline computation which means that there is no gain in performance. Unfortunately, the problem becomes NP hard for more than two dimension but has a polynomial-time logarithmic approximation.

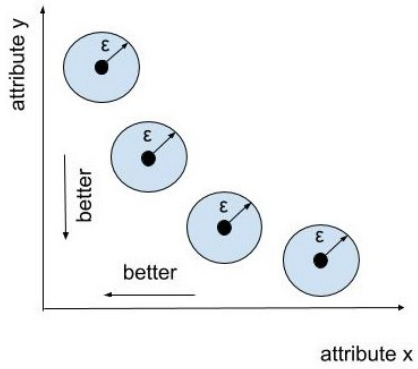


Figure 3.2: Approximate dominating representatives with ϵ distance

3.1.3. Weighting approaches

The goal of weighting approaches is again to produce a representative sample of the full skyline set just like summarization approaches, but their main difference is that they introduce ranking on skyline objects on some explicitly defined measure of interestingness. A common characteristic of this family of reduction algorithms is that objects with extreme values are discarded. There are mainly two categories of weighting approaches. The first is methods that are based on subspace analysis, so an object is more important when it appears on many subspaces. The second category ranks skyline objects based on how many objects they dominate.

As mentioned before on subspace analysis the notion of interestingness is the appearance of a skyline object in many subspaces. However, the computation of all subspaces is prohibitively expensive, so technologies have been developed to enable fast and efficient computation on subspaces. The most prominent of them is skycubes. Skycubes [52] maintain all skyline sets for all possible $2^d - 1$ subspaces of a database. Figure 3.3 depicts a skycube for attributes A, B, C, E. Given a lattice of subspace there are two main ways to compute the skyline set for all subspaces, a top down approach and a bottom up manner. In the bottom up approach the skyline set is obtained by merging the skyline sets of child nodes at the lower level. In the top down approach to determine the skyline set of a father node in the subspace lattice, the skylines of the child nodes are recursively enumerated. It is obvious that some skyline objects are reused and that makes top down approaches much more computationally efficient than bottom up approaches.

In [7] a metric called skyline frequency was introduced. That metric ranks skyline objects according to their number of appearances in all possible subspace skylines. A skyline object is more interesting the more times it appears in many subspace skylines. The k most interesting skyline objects are returned where k is user specified parameter. It can be seen that this metric incorporates ideas from both skyline and top-k queries,

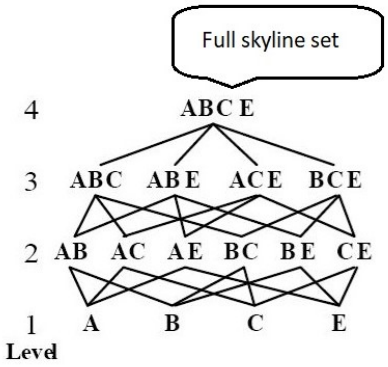


Figure 3.3: Example skycube for attributes A, B, C, E

but this time the preference function is the object's appearance in many subspaces. Skyline frequency allows for easy specification of skyline queries and the preference function is dependent on structural properties of subspace skylines. Figure 3.4 displays examples on top-1 and top-2 queries on a skyline set of objects a,b,d,j,k.

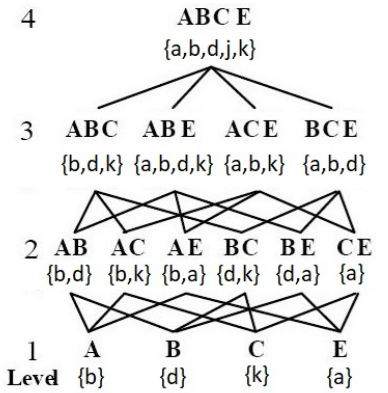


Figure 3.4: Example skycube for skyline points a,b,d,j,k on attributes A, B, C, E
Top-1 Frequent skyline={*b*}
Top-2 Frequent skyline={*b*, *a*}

Another algorithm belonging to this family of reduction algorithms is the Skyrank algorithm [48]. The Skyrank heuristic also relies on subspace relations, but this time a skyline object is more interesting the more objects it dominates on any subspace. The interestingness of a skyline point is propagated to all the skyline points that dominate it. To apply the Skyrank algorithm a skyline graph is necessary. A skyline graph can be constructed in the following way. The skyline objects are represented as nodes and edges are created in accordance with the skycube. For every skyline object it is tested whether it belongs to each subspace skyline in the skycube. If it is, no edge is created. If it is not, a directed edge is created from that skyline object to all objects which dominate that particular skyline object on that specific subspace. An example of a skyline graph is borrowed from [31] and is depicted on figure 3.6. The Skyrank algorithm can definitely be seen as similar to that of Pagerank. The basic heuristic of Pagerank is that a webpage is more interesting the more in-links it has, and also one webpage may propagate its importance to anther webpage. The skyrank algorithm applies link analysis similar to that of Pagerank in order to rank object, based on dominance relations.

The last subcategory of this kind of reduction algorithms are k-most representative skyline points [30]. This reduction algorithm aims at summarizing the skyline set by choosing the k most representative among them, where the representativeness in that case is determined by the number of objects a skyline point dominates. Again this methodology incorporates ideas from both top-k queries and approximate dominating

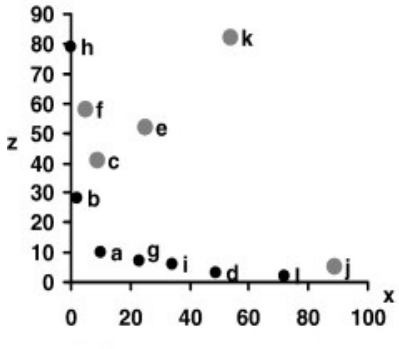


Figure 3.5: Skyline points on subspace $\{x, z\}$

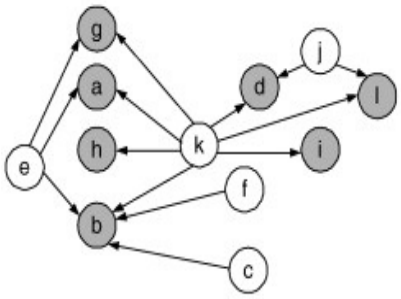


Figure 3.6: Skyline graph for subspace $\{x, z\}$

representatives, but this time the preference function is the dominance of the skyline object. Unfortunately, again the problem is np-hard for more than two dimension but can be approximated by greedy heuristics. This idea has been extended in [46] to retrieve the best k objects that minimize the distance between a non-representative skyline point and its nearest representative. The basic idea is that skyline points can be grouped into neighbourhoods of interestingness. The heuristic of distance based representatives tries to find a set of skyline points that best describe tradeoffs among these neighborhoods of interestingness. For example in our rooms data file it can be noticed that there exist three neighbourhoods of interestingness, shown in figure 3.7, N1 that contains very cheap rooms but far away from the city centre, N2 that contains very expensive rooms which are very close to the city centre, and N3 which is somewhere in between. Given an integer k, the distance base representative skyline set will contain k objects that best captures the contour of the full skyline set. So the DBR skyline set on the rooms data file for $k = 3$ will be $\{a, d, k\}$ as depicted in figure 3.8.

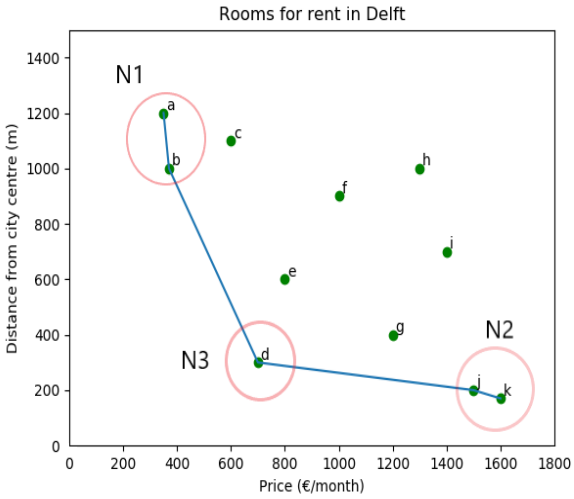


Figure 3.7: Neighbourhoods of interestingness

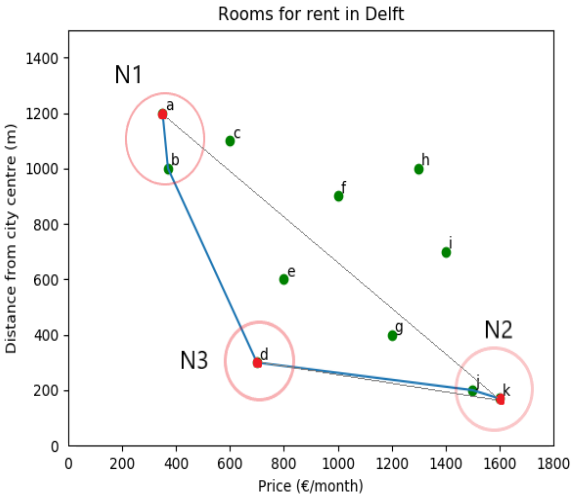


Figure 3.8: Distance based representative skyline set with $k = 3$, $S = \{a, d, k\}$

3.1.4. Cooperative approaches

All reduction algorithms presented up to now are based on structural properties in order to choose the most interesting points from the skyline sets. Cooperative approaches interactively try to elicit information in order to steer skyline reduction towards a more user preferable set. A framework proposed in [28] iteratively suggests the most informative preferences on partially ordered domains, because they are considered to be the reason that skyline sets end up being very large. The approach tries to minimize user interactions while minimizing the size of the skyline set.

Also trade off skylines [32] have been proposed that introduce trade offs between attribute domains. A

trade off can be viewed as a compensation between two attribute domains. For example a car dealer would ask a customer “Would you sacrifice 50hp for 5.000 euros?”. Trade off skylines are computationally expensive and that’s because trade offs modify the product order coming from attribute preferences. Also tradeoff relations are transitive and will probably introduce new ones resulting in a chain of tradeoffs.

3.2. Current evaluation methods

This section presents the methods that are used to evaluate already existing skyline algorithms.

In [3] a case study was conducted that computed and compared the sizes of the skyline sets and the running times of the Block Nested Loop algorithm(BNL) and the Divide and Conquer algorithm(DC) for different number of dimensions and for different types of databases, correlated databases where points that are good in one dimension are also good in the other dimensions, anti-correlated databases where points that are good in one dimension are bad in all others, and independent databases where there is no correlation between a point’s attribute values. Firstly, it was proved that for correlated databases the skyline set can become as large as 30% for 10 dimensions, and for anti-correlated databases it can become almost 75% of the database. Then, the running times for the skyline computation were measured for three variants of the BNL and the DC to reach the conclusion that the BNL is good if the size of the skyline set is small and there are no correlations in the database, while the DC algorithm is more resistant to the number of dimensions and correlations in the database.

In [26] where the Neighrest Neighbour algorithm(NN) was proposed, it was compared against four other already proposed skyline algorithms on the running time, the scalability, that is how the algorithm scales with different number of dimensions, again on correlated, anti-correlated and independent databases for 2-dimensional and multidimensional skylines. The results proved that the big advantage of the NN algorithm is that it shows a picture of the skyline set from the very first iterations. However, the research ends with the conclusion that there is no good and bad algorithm for computing the skyline set and every algorithm has its pros and cons.

In already existing research it can be seen that all reduction algorithms are not compared against each other, and that is highly logical since each author interprets interestingness in his own personal way. In [6] the authors proposed three variants of a skyline reduction algorithm. These three variants were evaluated and compared against each other on their running times, the scalability and the size of the reduced skyline set that they compute again for three types of databases. Correlated, anti-correlated and independent databases. Similarly, in [7] the proposed algorithm was implemented with two variants, one with precise counting and one with approximate counting. The two variants were compared against against each other on the running time and the scalability, on correlated, anti-correlated and independent databases.

In a similar way in [46], where the notion of Distance Based Representatives algorithm was proposed, again the variants of the algorithm presented were compared against each other on the running time and it was proved that the proposed algorithm better captures the contour of the entire skyline set, giving a more representative reduced skyline set from already existing summarization approaches.

4

Proposed way of using skyline sets

4.1. Introduction

This chapter proposes a new way of using skyline sets, that is to summarize datasets. This thesis follows a completely different approach to evaluate reduced skyline sets from the ones mentioned in the previous chapter. Two evaluations will be performed to evaluate reduced skyline sets for the new way of using them. The first one will evaluate reduced skyline sets based on the representativeness of the summarization they provide, by borrowing a methodology from statistical analysis. Then, the notion of summarization effectiveness will be redefined in a more user related manner and a case study will be designed to measure the summarization effectiveness of the reduction algorithms regarding a user related task.

4.2. Current applications

As mentioned previously, skyline algorithms have seen various applications in many fields such as market research, location based systems, bioinformatics. In existing applications skyline algorithms are used as an extension of traditional database systems to assist multi-criteria decision making systems, and to suggest users optimal objects, objects which are not dominated according to the notion of Pareto dominance. Figure 4.1 depicts the way skyline algorithms are used up to now. Taking for instance location based systems, one user might be interested in finding the shortest and the cheapest route to a particular destination. The skyline set of all the possible routes will be computed with the preference of minimizing both the distance and the price, these are the optimal routes, and only the skyline set will be returned to the user. Another example will be displayed on how a housing estate could use skyline algorithms to make suggestions. Suppose a user wants to find a room in Delft that is both cheap and close to the city centre. The housing estate could use the SFS algorithm analyzed in the background chapter to suggest rooms. So the rooms will be suggested in the following order $d, b, e, a, g, j, k, c, f, i, h$ as shown in table 2.2. A higher priority is given to rooms that will probably be more interesting to the user.

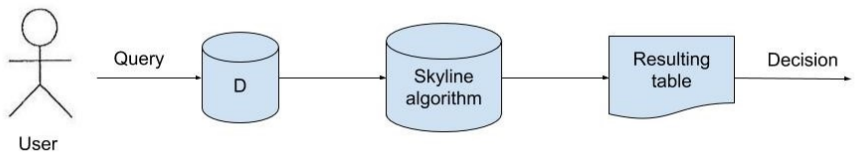


Figure 4.1: Current applications of skyline computation

4.3. Proposed framework

This thesis proposes a new way of using skyline sets, that is to summarize query results. Summarizing query results is defined similarly to that of summarizing databases [44], and is about choosing a set of tuples from query results in order to create a sample that still accurately reflects the diversity of attribute values of the original set. A framework is designed that proposes a way to integrate reduced skyline sets for interactive and progressive query refinement.

Refining queries has bothered the database research community for many years now and for many reasons. The main reason is mentioned in [37] and is that traditional relational database systems support mainly Boolean retrieval models on which constraints are applied on individual tuples and not on the resulting table. As a result, the cardinality of results cannot be reassured, and objects in resulting tables from SQL queries end up being too few or too many. In [37] an interactive framework was created which in order to satisfy cardinality constraints on query results, uses user feedback to reform user preferences.

Similarly, in [38] it is mentioned that the ranking of results from a user's query may not accurately reflect the preconceived notion of information need or similarity. In the same paper a query refinement system is designed that uses relevance feedback in order to handle user subjectivity in similarity search systems. To refine the query the system uses relevance feedback, where the user judges the relevance of the returned objects and the system refines the query to better reflect the user's information needs.

This master thesis proposes a new framework for interactive and progressive query refinement to assist filtering search on databases, but there are many more domains that it can be applied. The problem of filtering search has achieved great attention from the database research community. The goal of filtering search is to retrieve a set of records that would match a specific set of criteria. Filtering search is one of the main key features of SQL queries implemented by a "WHERE" clause. Also, filtering search [9] has seen many applications on already existing problems of computational geometry by improving their complexity and making them more simple to be implemented.

The framework is depicted in figure 4.2. The purpose of its design is to let users grasp an overview of the results right every query they perform. Given a user query two computations are performed. Firstly, the resulting table from the database(D) is computed, that is the "Resulting table". Then, the reduced skyline set is computed by the skyline reduction algorithm(RA) which receives as input the query and the full skyline set. The reduced skyline set, mentioned as "Overview", is returned back to the user and it will be used to summarize the results for that query. Then the user can get some ideas and refine his query towards any direction. Whenever the user feels ready the final decision will be made from the "Resulting table" from the database. The framework proposed was made progressive in the following way. Every query differs from its previous one by only one constrain. By just adding one filter to the query, it is dynamically submitted and the computation is repeated.

We see that our proposed framework for refining queries differs very much from the already existing ones. Although its structure looks similar to that of [37], it differs very much from that. Its purpose is not to engage users only when a cardinality constrain is not satisfied, but to progressively engage users by providing a feedback with the possible choices they have available.

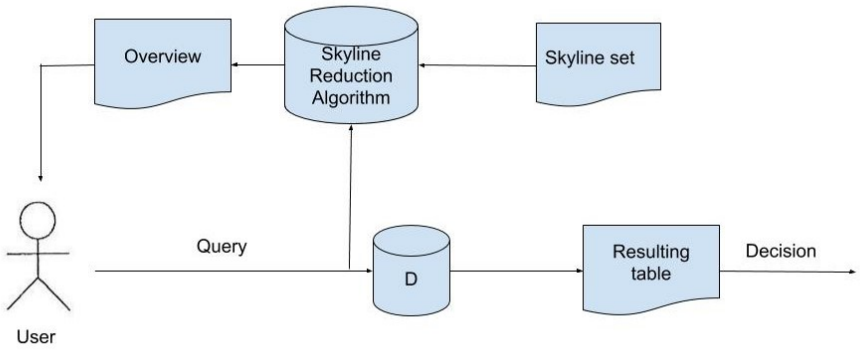


Figure 4.2: Architecture of the proposed framework

4.4. Evaluating a summarization

Summarization approaches have been studied extensively by the fields of data and knowledge mining. Automatic summarization[16] involves methods for finding a compact description for a subset of data. Many summarization approaches have been proposed, until now the simplest of which is displaying the mean and standard deviations for all fields. More complex summarization approaches require derivation of summary rules[2], multivariate visualization techniques, interactive exploratory data analysis[34].

Database summarization techniques have attracted particular interest from researchers. Database summarization [44] is the process of choosing a set of tuples from the database so that the resulting set is very much informative of the entire database, regarding the range of attribute values, but much smaller in size. There are three basic approaches to database summarization. The first of them is based on aggregate computation, such as statistical databases that compute aggregates by using statistical functions, and online analytical processing that uses cubes of computed aggregated values. The second is semantic compression. Semantic compression approaches produce high-level descriptions for a subset of database objects, some examples are association rules and decision trees. Finally, there are metadata-based semantic compression approaches that exploit meta-data to produce high level descriptions on a user defined vocabulary.

According to [22] the evaluation of a summarization method can be performed in an intrinsic or extrinsic manner. The intrinsic evaluation tests the summarization system on itself while the extrinsic evaluation tests the summarization based on how it affects the completion of a user related task such as providing useful information to find relevant documents. In the context of this master thesis reduced skyline sets will be evaluated in both of these ways.

4.5. Intrinsic evaluation

As discussed before it can be seen that a reduced skyline set summarizes the database since it chooses some tuples from it. One possible evaluation of reduced sets would be to determine the representativeness of every reduced skyline set towards the full skyline set. As mentioned in the second chapter, any user preference can be represented with a monotone function. One of the fundamental properties of the skyline set is that all objects that belong in the skyline set will maximize that function, which means that regardless how a user weights his personal preferences, the most favourable answer to every possible query will be found in the skyline set. And since every reduced skyline set is a subset of the full skyline set, the skyline set will be used as a reference point for all these comparisons.

In [41] it is mentioned that if no sampling error then the sample is representative of the population, so measuring the sampling error can show how much representative a sample of the database is. The sampling error describes the difference between the sampled statistic and the actual population parameter. In our case the sampled statistic is the reduced set means for every attribute, and the actual parameter is the full skyline set means for every attribute. The metric that will be used to measure the average sampling error per attribute between the full skyline set and a reduced skyline set is described in the following formula:

$$d(\textit{redset, skset}) = \frac{\sum_{i=1}^{d} |m_i - \mu_i|}{d}$$

where i iterates all dimensions d, m_i is the mean of the reduced set for attribute i and μ is the mean of the skyline set for attribute i. The findings of that metric will be presented in chapter 7.

4.6. Extrinsic evaluation

To evaluate the summarization provided by a reduction algorithm for the new proposed way of using reduced skyline sets, to summarize query results, by using the framework proposed before, a case study will be conducted that will evaluate how the summarization provided by a reduction algorithm affects the completion of a user related task. That task is going to be retrieving a relevant record according to a given description. After every attempted query users will get an overview of their possible choices and then they have to refine their query to get to their favourable answer.

Firstly, the notion of summarization needs to be redefined for the user experiment. The entire reduced skyline set will be provided to users and not just a statistic on that, and that summarization will regard the the overview of different choices the reduced set gives to the user. A summarization provided by a reduction algorithm will be evaluated on the following criteria in accordance with [22]:

- The user's success in accomplishing the task.

- The summarization's usefulness regarding the accomplishment of the task.
- The searching effort needed to perform the task on database by while using the summarization.

Many reduction algorithms(RAs) to summarize datasets will be tested in the context of this master thesis. The user experiment will tell the difference between those users who got a good overview, and will hopefully perform better, than those who didn't, and will not perform so good. In chapter 6 there will be a more detailed explanation of the experimental setup and all the variables that will be used to measure the summarization effectiveness as described above.

5

Clustering Based Skyline frequency

5.1. Introduction

One of the contributions of this master thesis is to propose a new reduction algorithm that adopts characteristics from both summarization and weighting approaches, the Clustering Based Frequency algorithm. Although clustering techniques have been studied extensively, no one has ever proposed using clustering methods to choose objects from the skyline set. The basic idea is to define neighbourhoods of interestingness by clustering points, that is to summarize the full skyline set, and then to apply subspace analysis to each cluster, that will choose the most meaningful objects from each cluster.

5.2. Motivation

The idea of Clustering Based Frequency was inspired from the Distance based representatives described in 3.1.3. It can easily be noticed that although neighborhoods of interestingness are mentioned, in the Distance Based Representatives algorithm there is not a specific definition of them. Also, up to now there have been proposed summarization approaches that do not introduce ranking between objects and simply rely on distance based measures, and on the other hand there are weighting approaches that usually neglect summarization and mainly focus on ranking objects. The Clustering Based Frequency strictly defines these neighborhoods by first clustering objects, and then it choose the top-k dominant objects from each cluster by applying the heuristic of skyline frequency. The heuristic of the skyline frequency within each cluster will probably lead to a more effective semantic summarization of the skyline set.

5.3. Clustering

Because people today face huge volumes of data in their everyday lives, the need to store, access, represent and analyze data has become crucial nowadays. One of the prominent techniques to summarize data is to classify, or simply group objects into similar categories. Classification systems can be divided in two big categories [51]. There are supervised learning systems that classify objects into a set of supervised classes, and unsupervised learning systems, that don't use any labelled data to assign objects and try to discover hidden structures in data. Unsupervised learning, or simply clustering systems, can be separated in two categories, partitioning clustering systems, where a set of data objects are grouped in a known number of distinct and non-overlapping clusters, and hierarchical systems where objects are grouped in a set of nested clusters organized as a tree. Also, as mentioned in the same research, to handle large scale datasets, density based clustering algorithms have been developed that group objects into a cluster once they have detected a dense neighbourhood of them.

5.4. Clustering Based Frequency

The Clustering Based Frequency algorithm consists of two phases, the first is clustering skyline objects, in order to summarize them, and then from each cluster apply the heuristic of the skyline frequency to choose the most dominant skyline objects. As for the first phase, since there are plenty of clustering algorithms to choose from, two variations will be proposed, one with the K-Means, and one with a density based clustering

algorithm, the HDBSCAN. Which variation is more useful depends to the particular application and both the advantages and disadvantages of each algorithm have to be taken into account.

- k-Means: The advantage of the k-Means algorithm is that it is easy to implement and fast to compute. The disadvantages are that it will produce very much tight clusters, and the number of clusters k to be given to the algorithm is very much difficult to be predict. The complexity of the algorithm $O(kdn)$ where d is the number of dimensions and n is the number of objects.
- HDBSCAN: The HDBSCAN like any density based clustering algorithm is able to cluster objects according to their density. The HDBSCAN was introduced in [4], in order to improve the DBSCAN algorithm. It is mentioned in the same paper that the DBSCAN algorithm produces flat clusters which are very much dependent on the choice of parameter ϵ, which means that DBSCAN fails to cluster objects of varying density. The HDBSCAN algorithm works with only one input parameter, Minpts, the minimum number of points that can make a cluster. However, the advantages of HDBSCAN come with an increased complexity of $O(dn^2)$ where d is the number of dimensions and n is the number of points.

Two variations of the algorithm will be presented one that that uses the k-means clustering algorithm, the KMFREQ algorithm, and one that uses the HDBSCAN, the DBFREQ algorithm.

The second step of the algorithm is common for both variations. The heuristic of the skyline frequency [7] will be applied to find the most dominant objects from each cluster. According to the skyline frequency, an object is more dominant when it is returned by many skyline sets when considering different number of dimensions, so the skyline frequency of a point can be found by making $2^d - 1$ distinct skyline queries for each non empty set of attributes. The higher the skyline frequency is the more interesting the point is since it is dominated on a fewer number of dimensions. Because it is very computational expensive to perform these $2^d - 1$ skyline queries, the authors managed to find a workaround by using approximated counting of subspaces.

5.5. Evaluation

The two variations will be compared against each other on the running time and the dimensionality effect on a synthetic dataset, and against the DBR algorithm on the summarization produced for different number of input parameters and returned skyline points on a realistic dataset. The code for the evaluation can be found in [47].

5.5.1. Effect of dimensionality

A synthetic data set of 1000 skyline points was produced. Figure 5.1 displays the running times of KMFREQ, on different number of clusters and for different dimensions, and figure 5.2 displays the running time of DBFREQ on different number of *Minpts* and different number of dimensions.

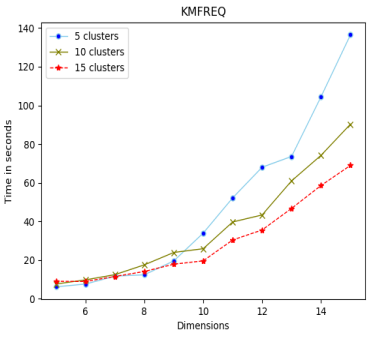


Figure 5.1: KMFREQ running times for different times

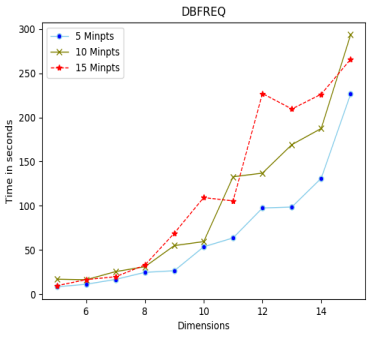


Figure 5.2: DBFREQ running times for different times

It can be seen that the KMFREQ algorithm is more efficient in time for all number of dimensions. The KMFREQ algorithm deteriorates for 5 clusters, and that is because when there are less clusters, there will be more objects within each cluster and the skyline heuristic will take much longer to finish. Exactly the same holds for DBFREQ. Whenever there are more dense clusters, clusters with many points in them, the algorithm will take longer to finish. Both algorithms scale pretty well for up to 11 dimensions.

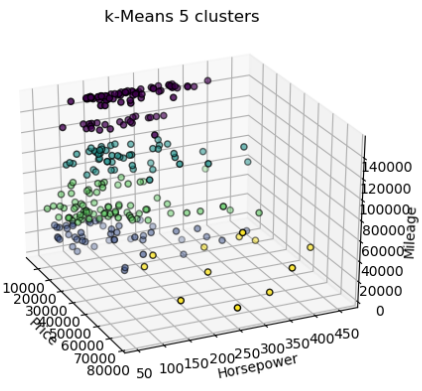
5.5.2. Summarization

A dataset of 2000 cars with three numerical attributes, price, horsepower, mileage, was used to evaluate the summarization produced by the new proposed algorithm. The original dataset can be found in kaggle [29]. The steps of preprocessing the dataset can found in detail in section 6.2. Figure 5.3 shows on the left column the clusters produced by the k-Means algorithm on the dataset for $k = 5, 10, 15$, and on the right column the HDBSCAN for $minpts = 15, 10, 5$.

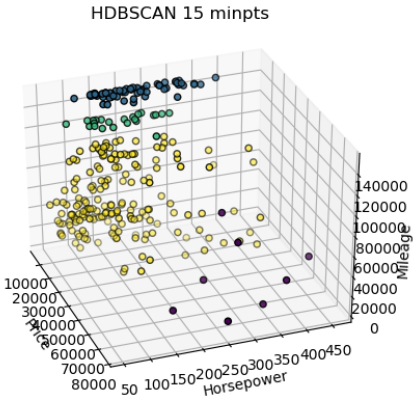
Figure 5.4 displays the points generated by the KMFREQ and DBR algorithms for $k = 5, 10, 15$. It can be noticed that for all k's the DBR algorithm summarizes the skyline set by choosing objects with extreme values which might not be that useful for answering queries on real time decision making systems. However, the KMFREQ algorithm chooses the most prominent object from each cluster, and because these points come from different clusters, that is different neighborhoods of interestingness, they still maintain the diversity of the full dataset. Exactly the same holds for the DBFREQ as shown in figure 5.5.

5.5.3. Discussion

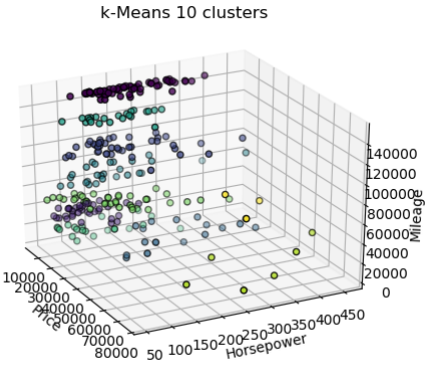
Current summarization approaches [31] such as the Distance Based Representatives algorithm [46], in order to represent the full skyline set rely on distance based measures to choose the most representative objects. However, these measures can choose extreme data points which might not be representative of the dataset and will probably lead users to deceptive assumptions about the dataset. They can also affect further statistical analysis as mentioned in [39]. The Clustering Based Frequency algorithm overcomes this problem by first clustering objects and then by applying the heuristic of skyline frequency which is guaranteed to find more meaningful objects from each cluster that will still describe the contour of the full skyline set.



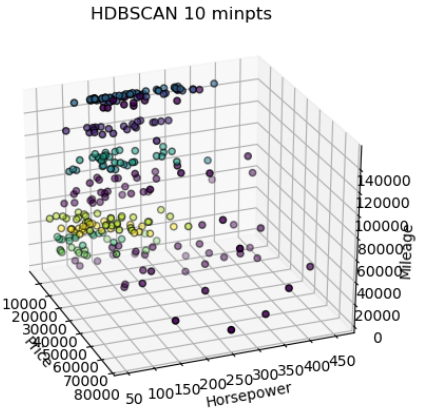
(a) k-Means with 5 clusters



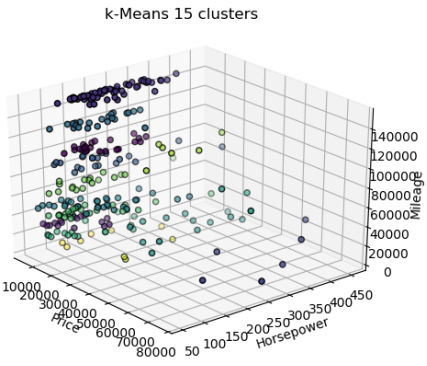
(b) HDBSCAN with 15 minpts produces 4 clusters



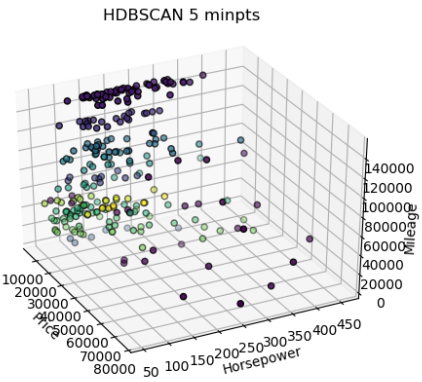
(c) k-Means with 10 clusters



(d) HDBSCAN with 10 minpts produces 10 clusters

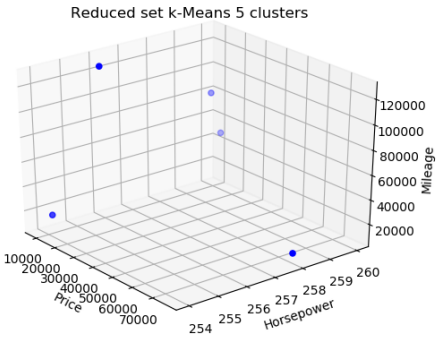


(e) k-Means with 15 clusters

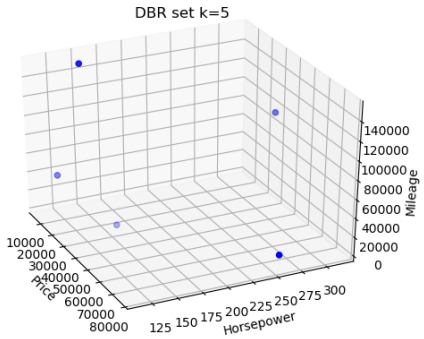


(f) HDBSCAN with 5 minpts produces 15 clusters

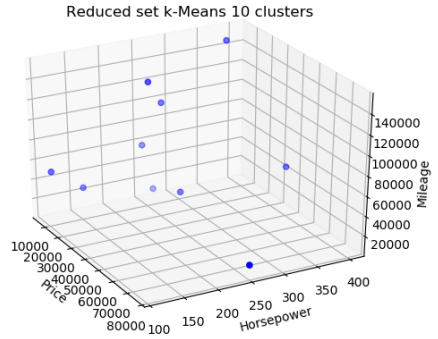
Figure 5.3: Clustering dataset points with k-Means and HDBSCAN



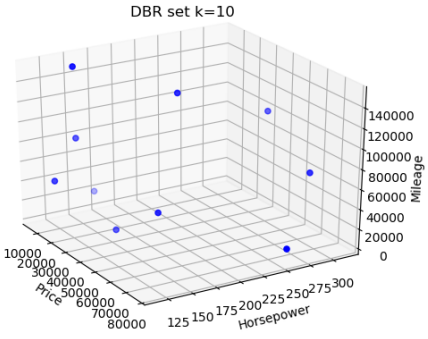
(a) KMFREQ with 5 clusters



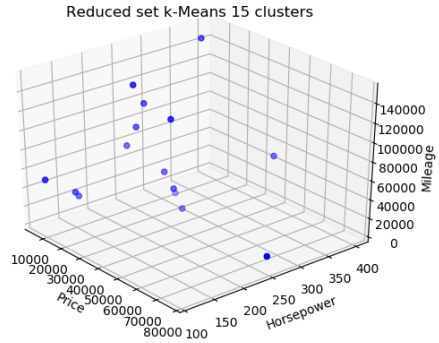
(b) DBR with k=5



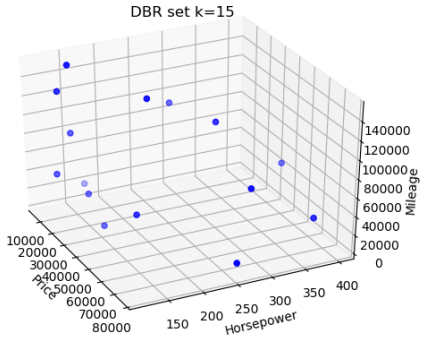
(c) KMFREQ with 10 clusters



(d) DBR with k=10

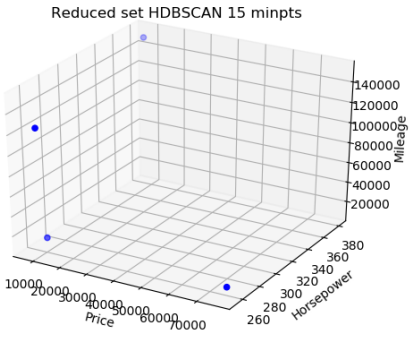


(e) KMFREQ with 15 clusters

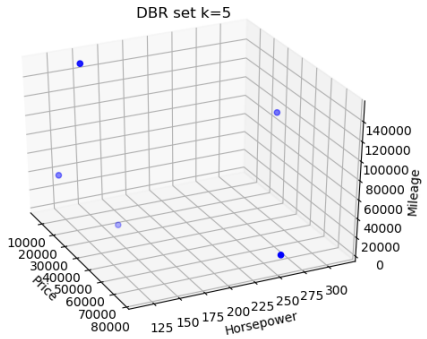


(f) DBR with k=15

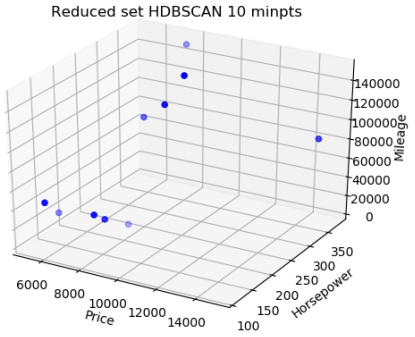
Figure 5.4: KMFREQ and DBR



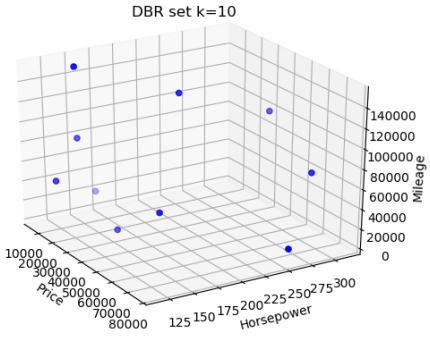
(a) DBFREQ with 15 minpts produces 4 clusters



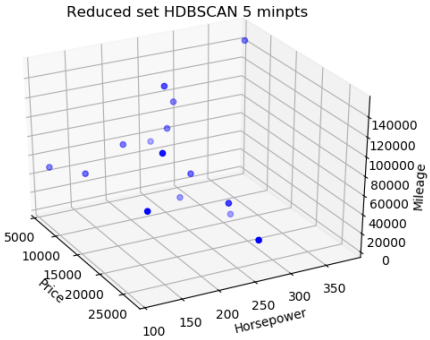
(b) DBR with k=5



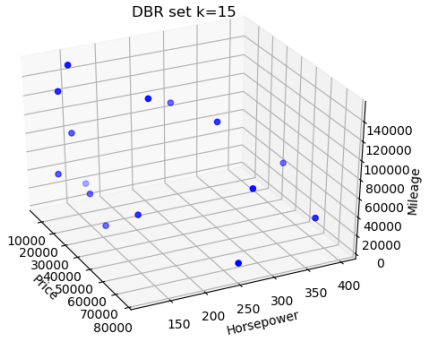
(c) DBFREQ with 10 minpts produces 10 clusters



(d) DBR with k=10



(e) DBFREQ with 5 minpts produces 15 clusters



(f) DBR with k=15

Figure 5.5: DBFREQ and DBR

6

Study setup

6.1. Case Study setup

6.1.1. Objectives

This case study proposes a framework for interactive query refinement to assist retrieval tasks on databases. The purpose of the case study is double-fold; firstly to evaluate whether providing users a summarization of their query results can help them refine their query and make better decisions while reducing the effort needed when performing retrieval tasks on databases, and secondly, to decide which skyline reduction algorithms would be the most useful for summarizing query results for both experienced and inexperienced users. So the concept of choosing the most dominant objects will be used to summarize query results. The research questions to be addressed are the following:

- Can reduction algorithms be used to summarize query results?
- Which skyline reduction algorithms are the most useful for summarizing query results?
- How the proposed framework affects the perceived satisfaction and effort of both experienced and inexperienced users on database retrieval tasks?

6.1.2. Background

The skyline operator [3] was proposed to fill the gap between traditional and multimedia database systems[15], and to provide personalized access to huge volumes of data. The purpose of the skyline operator is to choose optimal objects regarding a user preference. However, the skyline set becomes usually pretty large because of the curse of dimensionality [8]. Many reduction algorithms [31] have been proposed to choose interesting objects and reduce the size of the skyline set, but that notion of interestingness is really hard to formalize. Up to now skyline reduction algorithms are used as an extension of traditional database systems to extract interesting objects for users, but no one has ever examined one potential use of reduction algorithms, to summarize query results.

The problem of summarizing query results has bothered the research community for a long time and has seen great applications in web search engines. In [42] it is mentioned that users only look over the first 20 pages of returned results by a search engine. In both [43] and [42] a novel method of adaptive search was proposed, that uses clustering, summarization, and user feedback to assist interactive information seeking. The main idea is to group similar documents into clusters and to present only a summary of the cluster and then users are able to interactively expand their search by providing feedback. It is mentioned in both papers that this kind of search outperforms traditional querying search engines since it does not require knowledge on any Boolean querying language and keywords to describe a query.

Maximizing user satisfaction has been the main subject of recommendation systems. Basically, there are three kinds of recommendation systems [19]. Content based systems, which recommend items based on a user profile created at the beginning and an item's description, collaborative recommendation systems, which suggest items by collecting past ratings of all users, and hybrid systems, that can be seen as a combination of collaborative and content based filtering systems. It is mentioned in [19] that commercial systems

measure user satisfaction by the number of products purchased, while non commercial systems measure satisfaction by simply asking users how satisfied they are.

Interactive query refinement methods have been studied extensively for many different reasons, taking for example the interactive framework proposed in [37] that uses user feedback to reform user preferences in order to satisfy cardinality constraints on SQL query results. Also in [49] two methods are revisited for automatic query refinement for text based video retrieval, a two query expansion method based on pseudo-relevance feedback and a query refinement method based on text annotation. Furthermore, the problem of summarizing queries has seen great application in information retrieval to handle the word mismatch and to automatically expand queries using word relationships. A recent study in [50] compared all these methods on their effectiveness.

6.1.3. System's Overview

The system's architecture is illustrated in figure 4.2. This interactive query refinement framework was designed in order to assist filtering search on databases. After every attempted query the reduction algorithm computes the reduced skyline set for that query, and the reduced skyline set will be returned to the user and will give a summary of the query results. Then the user can get some ideas from that to refine the query. The final decision will be made from the resulting table from the entire dataset and not from the reduced skyline set, because if users were able to answer their query from the reduced set, then the use of the skyline set would be to answer queries directly as in already existing skyline applications. The main contribution of this master thesis is to use skyline sets to summarize query results and not to answer queries directly. The framework supports progressive query refinement because every query differs from its previous one by only one filter, that is one constrain. So, whenever the user adds a filter to the query, the query is dynamically submitted and both the resulting tables from the dataset and the reduction algorithm are re-performed.

6.1.4. Algorithms for summarization

Users were divided in 7 groups, with each group using one skyline algorithm to summarize query results. The different groups along with their respective algorithm are depicted on the table below.

Case	**Algorithm**
full	full skyline set
kdom	k-dominant skyline set
rand	random skyline set
skfreq	top-k frequent skyline set
dbfreq	density based frequent skyline set
dbr	distance based representative skyline set
free	no skyline set, users will work directly on the dataset

Table 6.1: Groups of users and skyline algorithm per group

Every reduced skyline set(kdom, rand, skfreq, dbfreq, dbr) was computed offline on the dataset by running the respective reduction algorithm. All the reduction algorithms that were used were already mentioned in chapter 3. To compute the k-dominant skyline set the two-scan algorithm was implemented as in [6], to compute the top-k frequent skyline set the algorithm outlined in [7] was used, the dbr skyline was computed by using the naive algorithm found in [46]. The full skyline set was computed offline too by using the block nested loop algorithm mentioned in section 2.5.1. The free group did not use any skyline set and got to work directly on the entire dataset. The random reduction is a skyline reduction algorithm that was implemented for the shake of this case study. The heuristic behind the algorithm is random sampling, the algorithm randomly chooses objects from the skyline set. The number of the skyline objects to be returned is an input to the algorithm. The dbfreq group used the density based frequency algorithm found in chapter 5. The code for all the computations can be found in [47].

6.1.5. Dataset for case study

The dataset consists of 2000 cars with 3 numerical and 4 categorical attributes. The dataset used for the study and the steps of pre-processing it will be analytically explained in section 6.2. Although the domain decided

for the experiment was the cars market exactly the same system is applicable to many other domains.

6.1.6. Participants

42 student from Tu Delft were randomly chosen and participated in the case study. An equal number of 6 participants were assigned to each algorithm, and since prior knowledge on the cars market can have an effect on user's performance, users were deliberately assigned into two user groups according to a pretest. From the 6 users in every algorithm, 3 users had some prior knowledge on the cars market, and 3 did not have. Each participant was assigned to only one group.

According to [33], young people(people between 18 and 34 years old) is the most prominent age group for purchasing cars, either directly, or indirectly, by advising their parents to do so. Thus, the findings of the case study can be generalized for the wider population.

6.1.7. Task

Each user was given a car's description found in appendix C.4 and had to search/filter the database to find a car that would best match that given description. The description is the same for all users.

6.1.8. Independent Variables

- Prior knowledge on the cars market: A self assessment method was created which can be found in appendix C.3. The variable has two levels, yes or no.

6.1.9. Dependent Variables

- Perceived mental effort: the mental effort the user devoted to the task.
- Total time: the total time since the user was given the description until the moment the final decision was made.
- Number of attempts: the total number of submitted queries.
- Satisfaction: the user satisfaction with the final decision.
- Usefulness: how much helpful was the proposed framework.

6.1.10. Blocking factors, noise factors and co-variances

The total time and the total number of attempted queries, are really difficult to compare for different users since each user will take his own time to re-filter the dataset to get to his favourable answer. Within each group there were two cases of users, because prior knowledge on the cars market will definitely affect the user's performance, experienced users and inexperienced users. Experienced users are the users that have some prior knowledge on the domain of the study and will probably not find the framework that useful, and inexperienced users that don't have any prior knowledge and will probably find the framework more useful.

6.1.11. Data sources and Instrumentation

The skyline set was computed offline with the BNL algorithm and it consists of 280 cars. Since the number of objects to be returned is an input to the reduction algorithms it was decided that all reduced skyline sets will be half the size of the skyline set, 140 objects. So 140 objects will try to summarize 2000 objects in the initial dataset except for the k-dominant skyline algorithm that returned only 10 objects. Still it was used because perhaps the users will find it very much useful during the task. When computing the skyline set with the dbfreq algorithm the *Minpts* set to 4, so 4 cars can make a cluster. After tests it was seen that this input parameter produced a lot of different clusters of interestingness.

This case study combines both qualitative and quantitative data from observations and questionnaires. A questionnaire was created found in appendix C. The questionnaire consists of two parts. The first part of the questionnaire can be found in C.1 and it regards the participant's personal details. The second part regards the evaluation of the algorithm that each user was assigned and it can be found in C.2. The questionnaire also consists of a self assessment method to evaluate the user's knowledge on the cars market, found in C.3, and according to that a user was inserted into the system as an experienced or inexperienced user.

A website was created to assist the case study [1], and all its code can be found in the following link [47]. The webserver was used to measure automatically the time spent on the task and the total number of at-

tempted queries, and the questionnaire was answered on the website as well. Also, a description of a car was created found in appendix C.4 on which the retrieval task was performed. Every user did the same task.

6.1.12. Procedure

Firstly, each user had to fill in the first part of the questionnaire with the personal details. Then, by doing the pretest it was decided whether the user is experienced or inexperienced and was placed in one of the 7 groups. After that, the user was given a brief overview of the system's architecture. Then, the user was given the car's description and was observed performing the retrieval task on the dataset in order to find a car that would best match that given description. At the end of the task, each user had to complete the second part of the questionnaire in order to evaluate the respective skyline algorithm.

6.1.13. Units of measurements

To evaluate the perceived mental effort of each user on the task and the perceived satisfaction, the Nasa Task Load Index protocol [18] was used that evaluates the perceived mental effort on a 7 point scale. The total time spent on the task is measured in seconds and the total number of querying attempts is an integer number.

6.1.14. Assumptions

- Perceived mental effort: in the free group(the group that users worked directly on the dataset) a big difference is expected between experienced and inexperienced users, in all other groups we expect a small difference between experienced and inexperienced users. So, the framework for summarizing query results should reduce the mental effort needed.
- Total time: will probably decrease when using the system, because the time to read the summarization is much less than going through a huge number of results.
- Number of attempts: will probably decrease when using the system, because the framework for query refinement guides users to their favourable answer.
- Satisfaction: it is expected that using skyline sets to summarize choices will probably increase user satisfaction, especially for inexperienced users. Perhaps the satisfaction will show which reduction algorithms are better for summarizing query results. The higher the satisfaction is, the better summarization the algorithm provides.
- Usefulness: will vary between skyline sets and will show which algorithms were the most helpful for summarizing choices. Perhaps inexperienced users will find the framework more useful.

6.2. Dataset preparation

6.2.1. Data cleaning

The dataset [29] used for the case study contains over 370000 used cars scraped with Scrapy from Ebay-Kleinanzeigen. The reference above contains a detailed description of the dataset attributes. From all these attributes, those that are not engaged in dominance relations were dropped out. The attributes that are kept and will be used for dominance relations are the following:

- price: The price in euros, lies between 5000 - 90000
- powerPS: Horsepower in PS, lies between 50 - 450
- mileage: How many kiloemeters the car's already run, lies between 5000 - 150000
- bodystyle: limousine, kombi, coupe, suv, bus, cabrio, wagon, other
- gearbox: automatic or manual
- fueltype: diesel, gas, cng, lpg, elektro, hybrid, other
- damage: yes if car has damage, else no

Next all the records containing at least one missing value were dropped out too, and still the remaining records were more than 100000. The next step was to replace the German descriptions in categorical data with English. To keep the size of the dataset manageable for the case study, a random sample of 2000 records was taken. These 2000 records form the final dataset that will be used for the case study.

6.2.2. Data binning

When comparing two objects for dominance relations it is very logical that a small difference in their values for the same attribute is not adequate to rule out an object. So, firstly the values will be placed into bins and then the dominance comparisons will be performed.

Numerical data To place numerical values into bins, a very flexible utility function was created to perform the binning process that works on the parameters shown in table 6.2.

Parameter	**Explanation**
Input parameters	
val	the value to be transformed
max	the maximum value for the attribute
min	the minimum value for the attribute
bin	the total number of bins for the attribute
invert	if true the assignment of bins will be inverted, so low values will be placed into higher bins
covered	if false an extra bin will be used for values above the maximum
Output parameter	
bin	he number of the bin that the input value was placed

Table 6.2: Utility function parameters

To make things clear an example will be explained for the attribute of price. The minimum value for price is 5000, the max was set to 25000, *covered* will be set to false because there are more expensive cars than 25000. The parameter *bins* is set to 4, so there are going to be 5 bins. Obviously the lower price is more preferable so the argument *invert* must be set to true. The bins created are the ones depicted in table 6.3

To make the problem look similar to that of vector maximization the higher value was decided to be more preferable over a lower one. Suppose we compare two price values $p_a = 7000$ and $p_b = 17000$ it can be seen that a price value of 7000 will be placed into bin 5 and a price value of 17000 will be placed into bin 3, $5 > 3$ so the value of 7000 is more preferable than 17000.

Similarly the bins for the attributes of horsepower and mileage were computed and are depicted in tables 6.4 and 6.5 respectively:

Price range	Bin
5000 – 10000	5
10000 – 15000	4
15000 – 20000	3
20000 – 25000	2
Over 25000	1

Table 6.3: Price bins

Horsepower range	Bin
50 - 100	1
100 - 150	2
150 - 200	3
200 - 250	4
above 250	5

Table 6.4: Horsepower bins

Mileage range	Output
0 – 50000	3
50000 – 100000	2
100000 – 150000	1

Table 6.5: Mileage bins

Categorical data To handle preferences over partially ordered attribute domains, a coding schema as in [5] is adopted where a mapping function f_i is maintained for every attribute domain D_i that maps every value $v \in D_i$ into an interval $N \times N$ where N is the set of natural numbers. For two attribute values v and $\bar{v}$ with $v \neq \bar{v}$, if $f(v)$ contains $f(\bar{v})$ then v dominates $\bar{v}$.

The encoding scheme works in the following way; A spanning tree is first constructed from the directed acyclic graph G_i, then the spanning tree is traversed on post order and each node is assigned a unique post order number. The interval of v, $f_i(v)$ is given by $[x, y]$ where $y = post(v)$ and x is the smallest post order number assigned to a descendant of v. The dominance condition now becomes the following: $f_i(v)$ contains $f_i(\bar{v})$ if there is a path from v to $\bar{v}$ in the spanning tree ST_i and vice versa. If $f_i(v)$ does not contain nor is contained in $f_i(\bar{v})$ then the two values are incomparable.

Since the intention was to make the computation of the skyline set as generic as possible because skyline sets are agnostic of preference functions, all values within one categorical attribute domain were made incomparable. Tables 6.6 and 6.7 show the intervals given to the attributes of bodystyle and gearbox respectively. For the attributes of fueltype and damage the intervals were computed in a similar way.

Bodystyle	Interval
coupe	(1,2)
cabrio	(3,4)
suv	(5,6)
wagon	(7,8)
limousine	(9,10)
bus	(11,12)
kombi	(13,14)
other	(15,16)

Table 6.6: Bodystyle intervals

Gearbox	Interval
manual	(1,2)
automatic	(3,4)

Table 6.7: Gearbox intervals

The skyline set on the dataset of 2000 cars contains 280 cars. Figure 6.2 displays the dataset on the left side and the skyline set on the right side. Unfortunately, it is not possible to depict categorical attribute values.

Github Link: The code for all the steps of prerocessing the dataset can be found in [47].

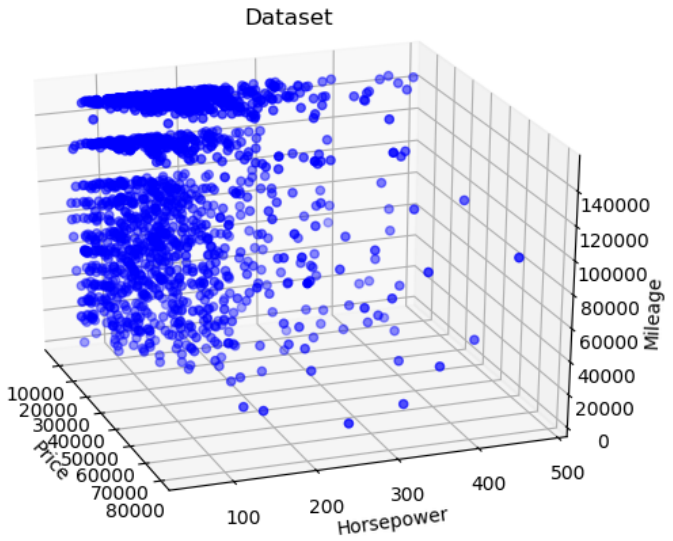


Figure 6.1: Dataset with 2000 cars

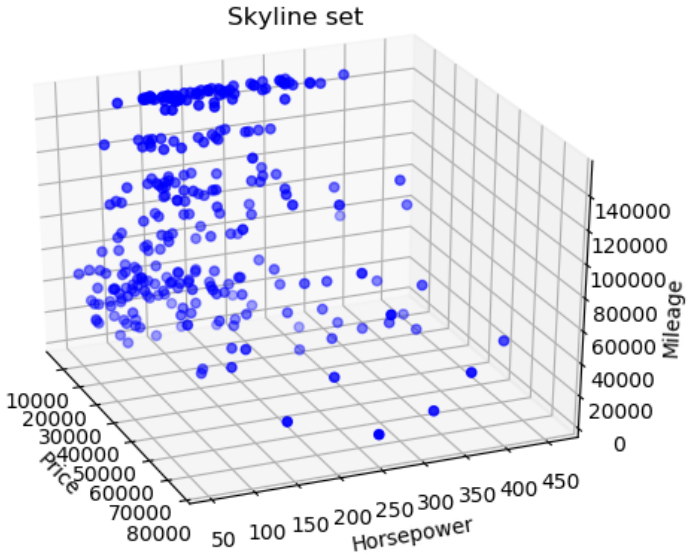


Figure 6.2: Skyline set

7

Findings

This chapter presents and discusses the results from both evaluations performed on skyline sets. Firstly, the findings on the representativeness will be elaborated. Then the results of the case study will be presented which will illuminate which algorithms are good for summarizing query results based on the new proposed framework for interactive query refinement.

7.1. Findings on representativeness

Table 7.1 displays the sampling error of the means for each skyline reduction algorithm. The sampling error shows the distance between the means of each reduced skyline set and the full skyline set. Since the dataset was normalized, all values were placed into bins to keep the computation of the skyline set fair as described in the previous chapter, the sampling error is measured in bins per attribute. The methodology was analytically explained in section 4.5. The smaller the sampling error the more representative of the dataset the reduced skyline set is as argued in the section mentioned before.

Algorithm	**Sampling error**
Random Reduction	0.12
Density Based Frequency	0.13
Distance Based Representatives	0.39
Skyline Frequency	0.42
k-dominant	0.61

Table 7.1: Sampling error of reduced skyline sets against the full skyline set

All algorithms manage to choose a very much representative sample from the skyline set, they all have a sampling error much smaller than one bin per attribute, which means that the calculated means from the reduced skyline sets are almost identical with those from the full skyline set. It can be seen that the algorithms that choose the most representative samples from the skyline set are the Random Reduction and the Density Based Frequency algorithms. Random Reduction manages to choose the most representative sample from the skyline set because the heuristic behind the algorithm is random sampling, which is guaranteed to minimize the sampling error and choose the most representative sample from the skyline set. The Density Based Frequency algorithm came second in the evaluation with a sampling error very close to that of Random reduction, so the heuristic of finding the best objects from every neighbourhood of interestingness chooses a very much representative sample from the skyline set. Then it is the Distance Based Representatives and the Skyline Frequency algorithms, and the last reduction algorithm in the ranking is the k-dominant skyline algorithm which still has a small sampling error.

As discussed before to evaluate reduction algorithms for the new way of using skyline sets, that is to summarize query results, the representativeness is just one interpretation of the summarization effectiveness. The findings of the case study next will complement those of the sampling error and will make clear which algorithms are the most useful for summarizing query results.

7.2. Findings of case study

This section presents the findings of the case study. The purpose of the case study was to propose a new framework for interactive query refinement where after every attempted query a summarization of the results was returned to users, and users had to refine their query to get to their preferable answer. Users were divided into 7 groups, where each group used a particular algorithm to summarize query results. Each group consisted of 6 users, 3 of them had some prior knowledge on the cars market, while 3 of them did not. Table 7.2 displays the results on the perceived difficulty, pressure, frustration and effort, while table 7.3 displays the results on the perceived usefulness, perceived satisfaction, total time spent and the number of attempts. In both tables the average(Avg) and the standard deviation(Std) are shown.

	Perceived Difficulty				**Perceived Pressure**				**Perceived Frustration**				**Perceived Effort**			
	Experienced		**Inexperienced**		**Experienced**		**Inexperienced**		**Experienced**		**Inexperience**		**Experienced**		**Inexperienced**	
	Avg	*Std*	*Avg*	*Std*	*Avg*	*Std*	*Avg*	*Std*	*Avg*	*Std*	*Avg*	*Std*	*Avg*	*Std*	*Avg*	*Std*
free	2.66	1.15	3.66	1.52	3.33	1.52	2.33	0.57	1.66	0.57	3	1.73	2.33	0.57	5	1
fullsk	1.66	0.57	2.33	0.57	1.66	1.15	2.66	1.15	1.66	0.57	1.33	0.57	2	1	2.33	0.57
dbfr	1.66	0.57	2	1	2.33	1.52	2	1.73	1	0	1	0	1.33	0.57	1.66	0.57
skfr	2.33	0.57	3.33	1.52	2.33	1.52	3	1	1.33	0.57	1.66	0.57	2.33	0.57	1.66	1.15
rand	1.66	0.57	2.33	1.15	1.16	0.57	3	1.73	1.66	0.57	1.33	0.57	2.33	1.52	2	1
kdom	2.66	1.15	2.33	0.57	2.66	2.08	2	1	1.33	0.57	2.33	1.53	2.66	1.15	2.33	0.57
dbr	2.66	0.57	3	2	1.66	0.57	2.66	1.52	1.33	0.57	1.66	0.57	2	1	3.33	2.51

Table 7.2: Results on perceived dificulty, pressure, frustration, effort

	Usefulness				**Perceived Satisfaction**				**Total time spent**				**Number of attempts**			
	Experienced		**Inexperienced**		**Experienced**		**Inexperienced**		**Experienced**		**Inexperience**		**Experienced**		**Inexperienced**	
	Avg	*Std*	*Avg*	*Std*	*Avg*	*Std*	*Avg*	*Std*	*Avg*	*Std*	*Avg*	*Std*	*Avg*	*Std*	*Avg*	*Std*
free	1	0	1	0	5.66	0.57	5	1	81.33	23.35	150.66	79.60	5	1.73	9	3.60
fullsk	6.33	0.57	5.66	0.57	6	1	5.33	1.15	121.66	6.02	141.33	82.08	5	1	6.33	4.04
dbfr	6	1	6	1.73	6	1	6	1	141.61	34.42	92.66	37.52	5.33	2.51	4.66	1.52
skfr	5.66	1.52	5.66	0.57	5.66	0.57	6	0	149.66	75.74	150.66	23.86	6.66	3.21	5	1
rand	6	0	6	0	5.66	0.57	5.66	0.57	199.33	76.74	115.33	46.5	7.66	5.50	4	1
kdom	4.33	2.87	3	2.65	6	0	3.66	2.08	154	68.02	178.33	119.52	5.33	1.15	14.33	9.07
dbr	5.33	1.52	5	2	6.33	0.57	6	1	174.33	68.80	153.33	47.52	9	2	7.33	3.21

Table 7.3: Results on perceived usefulness, satisfaction, time spent on the task and number of attempts

The average perceived usefulness per group is displayed in figure 7.1. Firstly, it can be noticed that users from all groups found the framework for summarizing query results very much useful for retrieval tasks on databases, and even experienced users had to say a good word. It can be seen that the k-dominant skyline algorithm was the least useful. Specifically, 3 users out of 6 reported that it was not helpful at all. All other skyline sets turned out to be very much useful for summarizing query results. It can be seen that the full skyline set was not the most useful of all as expected, since it is the largest set of all(all reduced skyline sets were half the size of the full skyline set). The data suggests that the usefulness of the full skyline set is almost in the same level with the usefulness of reduced skyline sets.

The hypotheses to be formulated are the following:

- Summarizing resulting tables from queries is very much useful for retrieval tasks on databases.
- The k-dominant skyline algorithm is not useful for summarizing query results.
- The full skyline set and the reduced skyline sets provided by the Distance Based Representatives, Skyline Frequency, Density Based Frequency, Random Reduction, are equally useful for summarizing query results.

The average satisfaction per group is displayed in figure 7.2. The first thing that can be noticed is that all users whether they used skyline sets to summarize query results or not, were very much motivated to reach their favourable answer. Experienced users in all groups were more satisfied with their final choice than inexperienced users. Inexperienced users in the free group(the group that worked directly on the dataset) and the kdom group(users that used k-dominant skyline to get an overview of their choices) were those that were less satisfied with their final choice. Inexperienced users that used the reduced skyline sets provided by the Skyline Frequency, Distance Based Representatives, Density Based Frequency and Random Reduction, to grasp an overview of their results were more satisfied with their final choice almost as much as experienced

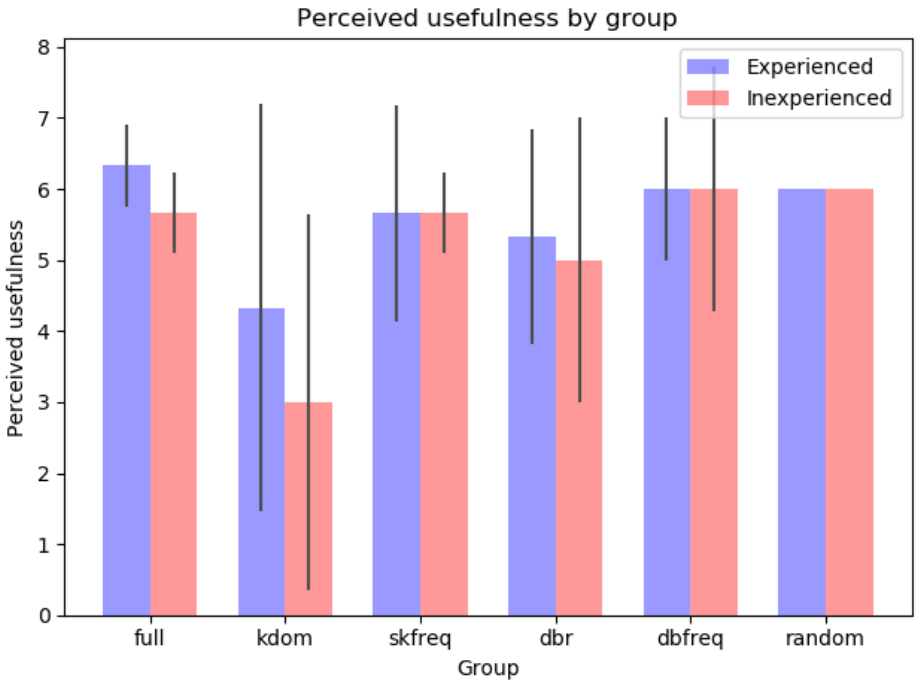


Figure 7.1: Average perceived usefulness per reduction algorithm

users in these groups. It can also be seen that there is a small difference in the perceived satisfaction between the full skyline set and reduced skyline sets except k-dominant skylines, so it can be assumed that they are all equally useful for maximizing user satisfaction.

The hypotheses to be formulated are the following:

- Using skyline algorithms to summarize query results helped increase inexperienced users' perceived satisfaction.
- The full skyline set and the reduced skyline sets provided by the Density Based Frequency, Distance Based Representatives, Random Reduction and Skyline Frequency are equally useful for improving user satisfaction.

With regard to the perceived effort, which is depicted in figure 7.3c, there is a correlation between the perceived effort and the perceived satisfaction. The harder a user tried the more satisfied he was with his final choice. So, the perceived satisfaction cannot be used to derive any conclusion about the summarization effectiveness of skyline reduction algorithms. It can be seen that inexperienced users in the free group had to spend much more effort during the task to get to their favourable answer, that's why they had a quite high level of satisfaction. On the other hand, inexperienced users in the kdom group had a low level of satisfaction and low effort as well. In all other cases that query results were summarized by skyline sets there is not such a big difference between experienced and inexperienced users. So we can conclude that using skyline algorithms to summarize choices helped decrease the perceived effort needed while increasing user satisfaction.

The formulated hypothesis is:

- Using skyline sets to summarize query results helped reduce the perceived effort of inexperienced users.

Concerning the perceived difficulty, depicted in figure 7.3a, it can be seen that although it seems that inexperienced users in the free group found the task more difficult, the perceived difficulty of inexperienced users in the free group is just a little bit higher than that of inexperienced users in the skfreq group. So, it can be concluded that the framework did not affect the perceived difficulty of inexperienced users. Exactly the same holds for experienced users.

The formulated hypothesis is:

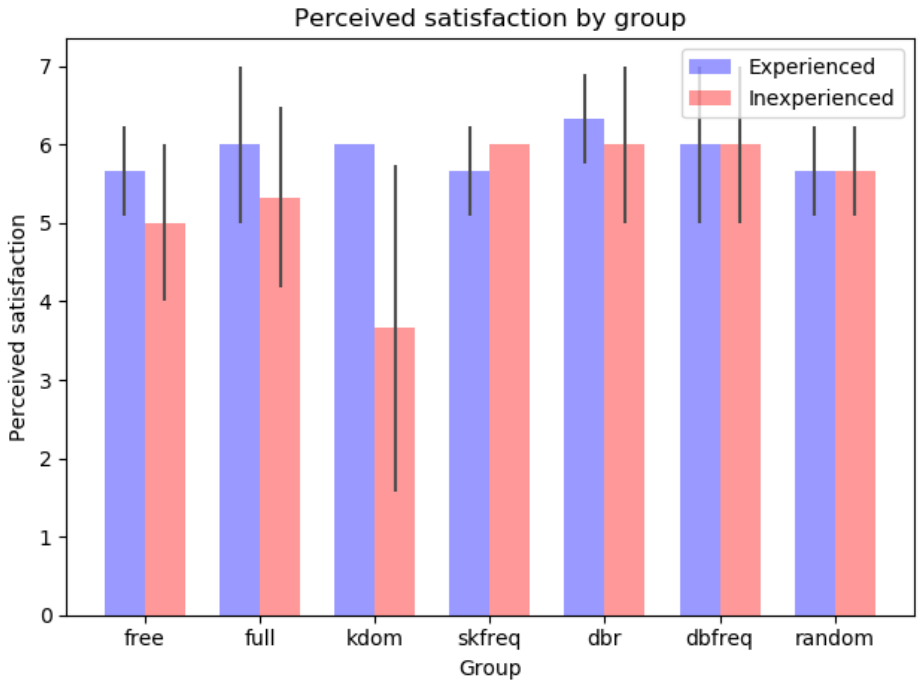


Figure 7.2: Average satisfaction per reduction algorithm

- Using skyline sets to summarize query results did not have an effect on the perceived difficulty.

Regarding the perceived pressure, depicted in figure 7.3b, no pattern can be extracted between groups. The formulated hypothesis is:

- Using skyline sets to summarize query results did not have an effect on the perceived pressure.

With regard to the perceived frustration, shown in figure 7.3d, it can be seen that users who used a skyline set, except k-dominant skyline set, achieved a lower frustration score. So it can be concluded that the framework supported users and reduced their anxiety.
The formulated hypothesis is:

- Using skyline sets to summarize query results helped reduce the anxiety of inexperienced users.

Regarding the total time spent on the task, results are depicted in table 7.3, it is not the case that in every group the total time spent by experienced users is smaller than that of inexperienced users as expected e.x in the rand group(the group that used random reduction to summarize query results) the average time of experienced users is 199 seconds while the average time for inexperienced users is 115 seconds. The only assumption that can be made is that experienced users in the free group spent less time on the task than experienced users in any other group. This is probably because the latter ones spent more time reading the summarization but again the times do not show evidence that this can turn out to be a significant difference. That leads to the following hypothesis:

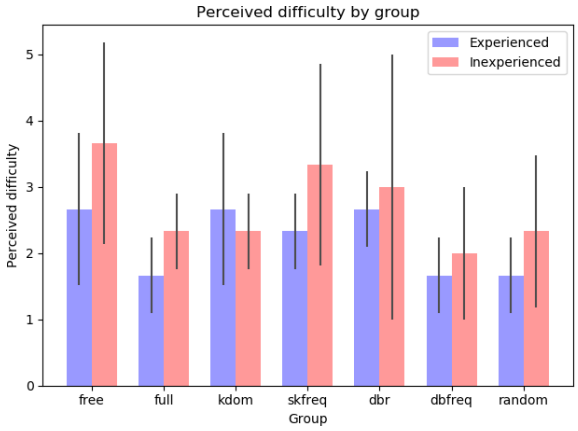
- Using skyline sets to summarize query results did not affect the time spent on the task.

Exactly the same holds for the total number of attempts until the final decision was made, depicted in table 7.3. No patterns can be extracted between and within groups.
The formulated hypothesis is:

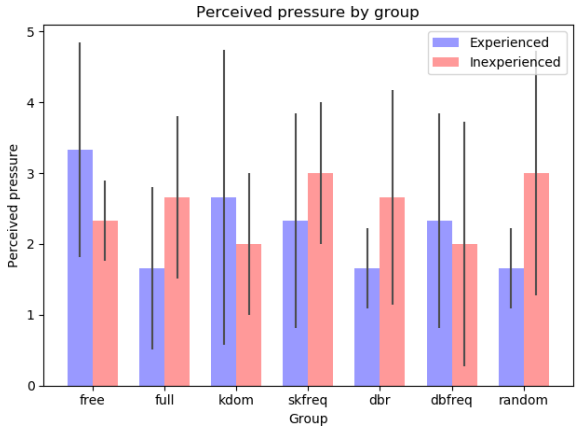
- Using skyline sets to summarize query results did not affect the number of attempted queries on the task.

7.3. Conclusions

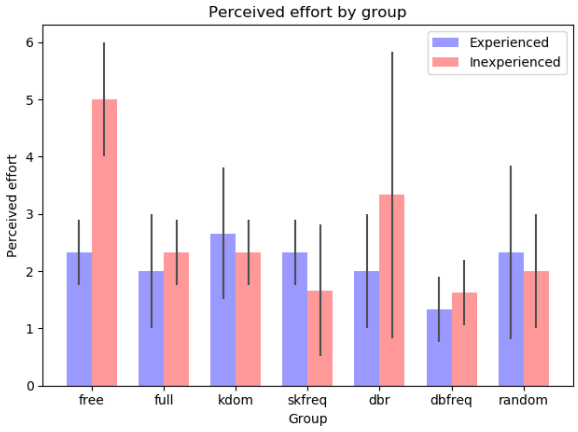
The data collected from the case study suggests that using reduction algorithms to summarize query results is very much useful for querying databases, especially for inexperienced users by increasing their satisfaction while decreasing the perceived effort needed. Also, the data suggests that the skyline set is equally useful with the reduced skyline sets provided by the four reduction algorithms, except the k-dominant skyline algorithm which cannot be used to summarize query results. Due to the limited number of participants this qualitative case study cannot determine significant differences. This is left as a goal for further quantitative research which will determine confidence intervals to create an actual ranking between reduction algorithms.



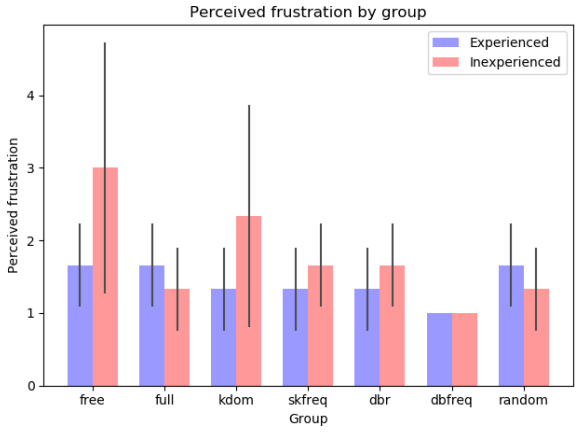
(a) Average perceived difficulty per reduction algorithm



(b) Average perceived pressure per reduction algorithm



(c) Average perceived effort per reduction algorithm



(d) Average perceived frustration per reduction algorithm

Figure 7.3: Perceived mental effort per reduction algorithm

8

Conclusion and future work

Skyline queries have attracted great attention and have seen many applications on real time decision making systems. This master thesis made the following contributions:

- A new way of using skyline sets was proposed, to summarize query results on real time multi-criteria decision making systems. A framework was created for interactive query refinement that used skyline sets to summarize query results in order to give users an overview of different choices they have available.
- A new reduction algorithm was proposed that better captures the contour of the full skyline set and chooses more meaningful objects at the same time.
- Skyline reduction algorithms were compared against each other based on their representativeness to find out that they all choose a very much representative sample of the dataset, and that the Random Reduction and the Density Based Frequency are the algorithms that choose the most representative samples from the dataset.
- On the grounds of summarization effectiveness the data collected from the case study suggests two things. Firstly, that users found the proposed framework very much useful for querying databases, and the framework can increase user perceived satisfaction while reducing the perceived effort needed, especially for users that don't have any prior knowledge on the particular domain. Secondly, the data suggests that the full skyline set and the reduced skyline sets provided by the Distance Based representatives, Random Reduction, Skyline frequency and Density Based frequency are equally useful for summarizing query results, although the reduced skyline sets were half the size of the full skyline set. So, these reduced skyline sets can be used instead of the full skyline set to support decision making in real time database filtering search systems.

This thesis ends with some propositions for future research:

- Although the data suggests that the proposed framework is very much useful for increasing user perceived satisfaction while decreasing the perceived effort needed, it should be the subject for further quantitative research to prove significant differences between the algorithms' usefulness, and to measure quantitative changes in the perceived effort and satisfaction.
- Maximizing user satisfaction has long been the subject of recommendation systems. In content based filtering systems objects are suggested according to the similarity with a user profile. In collaborative recommendation systems objects are suggested according to what other users found appealing. In hybrid systems techniques from both approaches are used. Skyline recommendation systems have been studied but no one has ever examined how such systems would affect user satisfaction on multi-criteria decision making systems.

A

List of tables

2.1 Bitmap encodings of example data file 9
2.2 Presorting based on entropy computation 12
2.3 Original and dynamic coordinates 13

6.1 Groups of users and skyline algorithm per group 32
6.2 Utility function parameters 35
6.3 Price bins 36
6.4 Horsepower bins 36
6.5 Mileage bins 36
6.6 Bodystyle intervals 36
6.7 Gearbox intervals 36

7.1 Sampling error of reduced skyline sets against the full skyline set 39
7.2 Results on perceived dificulty, pressure, frustration, effort 40
7.3 Results on perceived usefulness, satisfaction, time spent on the task and number of attempts . . 40

B

List of figures

2.1 Partial order preferences on the attributes of price, mileage, body-style . . . 5
2.2 The BNL algorithm for an input data file of 9 objects. The next object to be processed is $p7$ . . . 8
2.3 2-way partitioning . . . 8
2.4 3-way partitioning . . . 8
2.5 The rooms data file along with its skyline set(the points connected with blue line) . . . 9
2.6 Partitioning space with skyline point d . . . 10
2.7 Partitioning space with skyline points d, b, j . . . 10
2.8 Minimum bounded rectangles . . . 11
2.9 R-tree on the data file . . . 11
2.10 Constrained skyline query, $S = b, d$ . . . 12
2.11 Skyline query with constraints, $S = b, d$ . . . 12
2.12 Dynamic data space . . . 13
2.13 Dynamic skyline set . . . 13

3.1 Full skyline set and k-dominant skyline sets . . . 15
3.2 Approximate dominating representatives with ϵ distance . . . 16
3.3 Example skycube for attributes A, B, C, E . . . 17
3.4 Example skycube for skyline points a,b,d,j,k on attributes A, B, C, E Top-1 Frequent skyline=$\{b\}$ Top-2 Frequent skyline=$\{b, a\}$ . . . 17
3.5 Skyline points on subspace $\{x, z\}$ . . . 18
3.6 Skyline graph for subspace $\{x, z\}$ . . . 18
3.7 Neighbourhoods of interestingness . . . 18
3.8 Distance based representative skyline set with $k = 3$, $S = \{a, d, k\}$ . . . 18

4.1 Current applications of skyline computation . . . 21
4.2 Architecture of the proposed framework . . . 22

5.1 KMFREQ running times for different times . . . 26
5.2 DBFREQ running times for different times . . . 26
5.3 Clustering dataset points with k-Means and HDBSCAN . . . 28
5.4 KMFREQ and DBR . . . 29
5.5 DBFREQ and DBR . . . 30

6.1 Dataset with 2000 cars . . . 37
6.2 Skyline set . . . 37

7.1 Average perceived usefulness per reduction algorithm . . . 41
7.2 Average satisfaction per reduction algorithm . . . 42
7.3 Perceived mental effort per reduction algorithm . . . 43

C

Questionnaire

C.1. Personal details

- What is your age group?
    1. Child(Up to 14 years old)
    2. Youth(15-24 years old)
    3. Adult(25-64 years old)
    4. Senior(65 years old)
- What is your gender?
    1. Male
    2. Female
- What is your educational level?
    1. No formal education</option>
    2. Primary education
    3. Secondary education or high school
    4. GED
    5. Vocational qualification
    6. Bachelor's degree
    7. Master's degree
    8. Doctorate or higher
- Do you have any prior knowledge on the cars market?
    1. Yes
    2. No

C.2. Algorithm evaluation

- How mentally demanding was the task?
- How hurried or rushed was the pace of the task?
- How satisfied are you with your final decision?
- How hard did you have to work to complete the task(your searching effort)?
- How discouraged, irritated, stressed and annoyed were you?
- How helpful was the overview in making your decision(if you used it)?
- Do you have any comments to make(perhaps about the reduced set)?

C.3. Self assessment method

- How familiar are you with the cars market?
    1. I have never searched a cars marker before
    2. have searched a car market before
- How often do you search the cars market?
    1. Never
    2. More than once in a week
    3. More than once in a month
    4. More than once in a year
- How confident are you in searching the cars market?
    - Not all
    - Not very much
    - Enough
    - Very much
- Given some car attribute values(e.x the horsepower, mileage) could you make any sense of them?
    1. I have never heard of them or I have heard of them but don't know what they are.
    2. I have some idea what they are, but don't know when or how to use them.
    3. I have a clear idea what they are, but haven't used them.
    4. I can explain what they are and what they do, and I have used them.

C.4. Task description

Search for a very fast, new and cheap car of whatever bodystyle you want

Bibliography

[1] Reducing skyline sets. `http://reducesk.pythonanywhere.com/`. Accessed: 2019-02-24.

[2] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, A Inkeri Verkamo, et al. Fast discovery of association rules. *Advances in knowledge discovery and data mining*, 12(1):307–328, 1996.

[3] Stephan Borzsony, Donald Kossmann, and Konrad Stocker. The skyline operator. In *Data Engineering, 2001. Proceedings. 17th International Conference on*, pages 421–430. IEEE, 2001.

[4] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer, 2013.

[5] Chee-Yong Chan, Pin-Kwang Eng, and Kian-Lee Tan. Stratified computation of skylines with partially-ordered domains. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 203–214. ACM, 2005.

[6] Chee-Yong Chan, HV Jagadish, Kian-Lee Tan, Anthony KH Tung, and Zhenjie Zhang. Finding k-dominant skylines in high dimensional space. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 503–514. ACM, 2006.

[7] Chee-Yong Chan, HV Jagadish, Kian-Lee Tan, Anthony KH Tung, and Zhenjie Zhang. On high dimensional skylines. In *International Conference on Extending Database Technology*, pages 478–495. Springer, 2006.

[8] Surajit Chaudhuri, Nilesh Dalvi, and Raghav Kaushik. Robust cardinality and cost estimation for skyline operator. In *null*, page 64. IEEE, 2006.

[9] Bernard Chazelle. Filtering search: A new approach to query-answering. *SIAM Journal on Computing*, 15(3):703–724, 1986.

[10] Lei Chen and Xiang Lian. Dynamic skyline queries in metric spaces. In *Proceedings of the 11th international conference on Extending database technology: Advances in database technology*, pages 333–343. ACM, 2008.

[11] Jan Chomicki, Parke Godfrey, Jarek Gryz, and Dongming Liang. Skyline with presorting: Theory and optimizations. In *Intelligent Information Processing and Web Mining*, pages 595–604. Springer, 2005.

[12] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[13] Bin Cui, Hua Lu, Quanqing Xu, Lijiang Chen, Yafei Dai, and Yongluan Zhou. Parallel distributed processing of constrained skyline queries by filtering. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 546–555. IEEE, 2008.

[14] Evangelos Dellis and Bernhard Seeger. Efficient computation of reverse skyline queries. In *Proceedings of the 33rd international conference on Very large data bases*, pages 291–302. VLDB Endowment, 2007.

[15] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. *Journal of computer and system sciences*, 66(4):614–656, 2003.

[16] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37, 1996.

[17] Sven Ove Hansson. Preference logic. In *Handbook of philosophical logic*, pages 319–393. Springer, 2001.

[18] Sandra G Hart and Lowell E Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Advances in psychology*, volume 52, pages 139–183. Elsevier, 1988.

[19] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[20] Xuegang Huang and Christian S Jensen. In-route skyline querying for location-based services. In *International Workshop on Web and Wireless Geographical Information Systems*, pages 120–135. Springer, 2004.

[21] Ihab F Ilyas, George Beskales, and Mohamed A Soliman. A survey of top-k query processing techniques in relational database systems. *ACM Computing Surveys (CSUR)*, 40(4):11, 2008.

[22] MANI Inderjeet. Summarization evaluation: an overview. In *Proceedings of the NTCIR Workshop*, volume 2, 2001.

[23] Jonas Jarvius and Ulf Landegren. Dna skyline: fonts to facilitate visual inspection of nucleic acid sequences: Letter to the editor. *Biotechniques*, 40(6):740, 2006.

[24] Christos Kalyvas and Theodoros Tzouramanis. A survey of skyline query processing. *arXiv preprint arXiv:1704.01788*, 2017.

[25] Vladlen Koltun and Christos H Papadimitriou. Approximately dominating representatives. In *International Conference on Database Theory*, pages 204–214. Springer, 2005.

[26] Donald Kossmann, Frank Ramsak, and Steffen Rost. Shooting stars in the sky: An online algorithm for skyline queries. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 275–286. VLDB Endowment, 2002.

[27] Hsiang-Tsung Kung, Fabrizio Luccio, and Franco P Preparata. On finding the maxima of a set of vectors. *Journal of the ACM (JACM)*, 22(4):469–476, 1975.

[28] Jongwuk Lee, Gae-won You, Seung-won Hwang, Joachim Selke, and Wolf-Tilo Balke. Optimal preference elicitation for skyline queries over categorical domains. In *International Conference on Database and Expert Systems Applications*, pages 610–624. Springer, 2008.

[29] Orges Leka. Used cars database. URL `https://www.kaggle.com/orgesleka/used-cars-database/version/1`.

[30] Xuemin Lin, Yidong Yuan, Qing Zhang, and Ying Zhang. Selecting stars: The k most representative skyline operator. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 86–95. IEEE, 2007.

[31] Christoph Lofi and Wolf-Tilo Balke. On skyline queries and how to choose from pareto sets. In *Advanced Query Processing*, pages 15–36. Springer, 2013.

[32] Christoph Lofi, Ulrich Güntzer, and Wolf-Tilo Balke. Efficient computation of trade-off skylines. In *Proceedings of the 13th international conference on extending database technology*, pages 597–608. ACM, 2010.

[33] Ingo Lütkebohle. The Young and the Carless? The Demographics of New Vehicle Purchases. `https://www.federalreserve.gov/econresdata/notes/feds-notes/2016/the-young-and-the-carless-the-demographics-of-new-vehicle-purchases-20160624.html`, 2016. [Online; accessed 05-Dec-2019].

[34] Sergey Malinchik, Belinda Orme, Joseph A Rothermich, and Eric Bonabeau. Interactive exploratory data analysis. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 1, pages 1098–1104. IEEE, 2004.

[35] Duncan McCallum and David Avis. A linear algorithm for finding the convex hull of a simple polygon. *Information Processing Letters*, 9(5):201–206, 1979.

[36] J Mckendrick. Managing the rapid rise in data growth: 2011 ioug survey on database manageability. unisphere research, a division of information today, inc, 2011.

[37] Chaitanya Mishra and Nick Koudas. Interactive query refinement. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, pages 862–873. ACM, 2009.

[38] Michael Ortega-Binderberger, Kaushik Chakrabarti, and Sharad Mehrotra. An approach to integrating query refinement in sql. In *International Conference on Extending Database Technology*, pages 15–33. Springer, 2002.

[39] Jason W Osborne and Amy Overbay. The power of outliers (and why researchers should always check for them). *Practical assessment, research & evaluation*, 9(6):1–12, 2004.

[40] Dimitris Papadias, Yufei Tao, Greg Fu, and Bernhard Seeger. An optimal and progressive algorithm for skyline queries. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 467–478. ACM, 2003.

[41] Charles A Ramsey and Alan D Hewitt. A methodology for assessing sample representativeness. *Environmental Forensics*, 6(1):71–75, 2005.

[42] Dmitri G Roussinov and Hsinchun Chen. Information navigation on the web by clustering and summarizing query results. *Information Processing & Management*, 37(6):789–816, 2001.

[43] Dmitri G Roussinov and Michael J McQuaid. Information navigation by clustering and summarizing query results. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, pages 9–pp. IEEE, 2000.

[44] Régis Saint-Paul, Guillaume Raschia, and Noureddine Mouaddib. General purpose database summarization. In *Proceedings of the 31st international conference on Very large data bases*, pages 733–744. VLDB Endowment, 2005.

[45] Kian-Lee Tan, Pin-Kwang Eng, Beng Chin Ooi, et al. Efficient progressive skyline computation. In *VLDB*, volume 1, pages 301–310, 2001.

[46] Yufei Tao, Ling Ding, Xuemin Lin, and Jian Pei. Distance-based representative skyline. In *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*, pages 892–903. IEEE, 2009.

[47] K Touloumis. Skyline queries. `https://github.com/ktouloumis/Skyline-Queries`, 2018.

[48] Akrivi Vlachou and Michalis Vazirgiannis. Ranking the sky: Discovering the importance of skyline points through subspace dominance relationships. *Data & Knowledge Engineering*, 69(9):943–964, 2010.

[49] Timo Volkmer and Apostol Natsev. Exploring automatic query refinement for text-based video retrieval. In *Multimedia and Expo, 2006 IEEE International Conference on*, pages 765–768. IEEE, 2006.

[50] Jinxi Xu and W Bruce Croft. Quary expansion using local and global document analysis. In *Acm sigir forum*, volume 51, pages 168–175. ACM, 2017.

[51] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16 (3):645–678, 2005.

[52] Yidong Yuan, Xuemin Lin, Qing Liu, Wei Wang, Jeffrey Xu Yu, and Qing Zhang. Efficient computation of the skyline cube. In *Proceedings of the 31st international conference on Very large data bases*, pages 241–252. VLDB Endowment, 2005.