Universiteit Leiden
Opleiding Informatica

Systems of multi-criteria decision making
for Skyline Queries

Name: Wahagn Mkrtchyan
Date: 22/08/2019
1st supervisor: Michael Emmerich
2nd supervisor: Kaifeng Yang

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Contents

1 Abstract

Given that there are dominating items in data, a Skyline query gives back the items that have no item in the data that is dominating them. If there is more than one criteria, an object is not dominated by any other object if it is, better overall than any other object, this means to have at least as high or low values, depending on if the criteria represents a positive or negative characteristic, in all fields as any of the other items and one or more values that it excels in compared to any of the other entries, for all fields used in the skyline process.

2 Research Question

How can we use relational databases to support (skyline) queries in such a way that a system can be implemented that ranks solutions if there are multi-criteria for decision making?

3 Workflow

Introduction

For my bachelor's thesis after some consultations with different professors I have landed on the topic of Skyline Queries. During the introductory talks with professor Emmerich it was decided to work on a web-based skyline query application and that the thesis would be the creation of a website that makes this skyline querying possible after which the process and mathematics behind the programming should be presented in this document, the theoretical part of the thesis. The requirements for the application were that a user can upload CSV file, choose maximum/minimum bounds and criteria after which the file is processed using these parameters and the skyline query is executed using simple SQL within a database. The result had to preferably be illustrated, in a graph or otherwise, and there should be a possibility for the user to download the same CSV file after the processing and computations with the ranks attached that the query produced.

Software and setup

During the exploration phase a decision has been made by me to create the system with Laravel as this web framework has the necessary qualities. It was important for the project to be able to

- Build the web application fast in a timely manner
- Have a fast website, meaning pages open quickly
- Create a good looking front-end
- Use a small amount of time to learn the framework

Laravel framewrok logo;

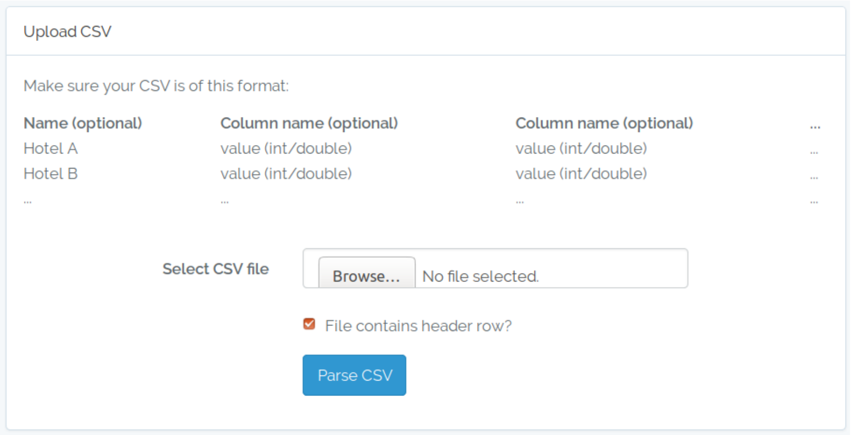
Laravel is fast, has a small learning curve and an easy to use front-end language called blade as well as good documentation.
On top of that it seemed a good idea to choose this widely used framework, because the popular software has more functions, libraries and packages especially user created ones that can be useful as well as for future career opportunities it might come in handy to have experience with this software used widely in web development.

Already having domains and hosting helped to quickly get everything set up. I have multiple domains at Vimexx and using the Installatron in the DirectAdmin installed Laravel version 5.1.11 on one of them and created a corresponding database, with phpMyAdmin using the default MySQL system and InnoDB as the storage engine.

Modern databases such as MySQL which was used in this application as well as others such as Microsoft SQL Server and Oracle are based on SQL which itself is based on RDBMS, the Relational database Management System. Entities with attributes and relations between the entities that depict the structure of the database. SQL stands for Structured Query Language and is the world's most popular language for controlling relational databases. The language was developed by IBM in the 1970s to invoke data from IBM's relational System R database system. Previous methods made it difficult to handle data from complex mainframes, which is why IBM started developing SEQUEL, as the language was then called. The name was later renamed to SQL for legal reasons. SQL is used for queries for this application because it's the standard language for creating, defining, altering and retrieving information in/from a database.

Application

The application has been programmed in such a way that at first an upload screen is shown where the user is asked to upload a CSV file and specify with a checkbox whether the file contains a header, meaning the first line contains the names of each of the columns. It is expected of the user that the CSV file with or without a header contains other than the header, lines of data all with the same amount of columns that contain the name of the entry in the first column and an integer or decimal number in the other columns.



Upload CSV

Make sure your CSV is of this format:

Name (optional)	Column name (optional)	Column name (optional)	...
Hotel A	value (int/double)	value (int/double)	...
Hotel B	value (int/double)	value (int/double)	...
...	...	...	...

Select CSV file: No file selected.

☑ File contains header row?

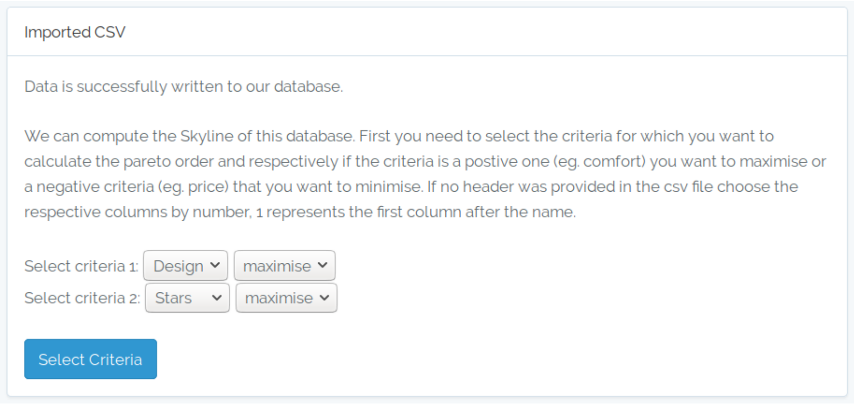
landing page, OurHolland.com (2019);

All those lines taken together, apart from the header line, of this file are then inserted with a JSON encoding to a field in a table, the header line is also JSON encoded and inserted in another field in the same entry. The same entry also contains a boolean that is set according to if the checkbox for header was true or not, a string with the name of the file provided by the user and a specification of the id of the entry.

All the data lines that are now in the data field in this table are decoded and inserted line for line column by column to a second table in the database, just as they were represented in the original CSV file. The column names are set according to the encoded header line in the other field of the first table or if no header was provided they are set numerically to _0, _1, _2 etc.

Parameters

The user is then asked to provide two criteria to use for the skyline process and whether each of these attributes need to be maximised or minimised. Hereafter the user can select bounds for the criteria selected on the previous page. If maximise was chosen this means that the minimum bound chosen is the lowest value acceptable for the respective criterion and the maximum bound the value after which further maximisation would yield no added value for the user. Vice versa for minimisation the maximum bound is the highest still acceptable value while the minimum bound specifies the border after which further minimisation would not lead to greater value for the user for that respective criterion.



Imported CSV

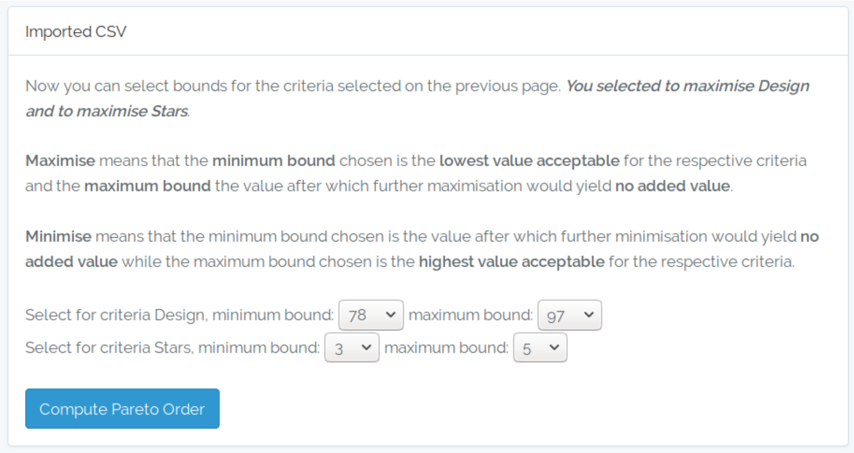
Data is successfully written to our database.

We can compute the Skyline of this database. First you need to select the criteria for which you want to calculate the pareto order and respectively if the criteria is a postive one (eg. comfort) you want to maximise or a negative criteria (eg. price) that you want to minimise. If no header was provided in the csv file choose the respective columns by number, 1 represents the first column after the name.

Select criteria 1: Design maximise

Select criteria 2: Stars maximise

import_process, OurHolland.com (2019);



Imported CSV

Now you can select bounds for the criteria selected on the previous page. ***You selected to maximise Design and to maximise Stars.***

Maximise means that the **minimum bound** chosen is the **lowest value acceptable** for the respective criteria and the **maximum bound** the value after which further maximisation would yield **no added value**.

Minimise means that the minimum bound chosen is the value after which further minimisation would yield **no added value** while the maximum bound chosen is the **highest value acceptable** for the respective criteria.

Select for criteria Design, minimum bound: 78 maximum bound: 97
Select for criteria Stars, minimum bound: 3 maximum bound: 5

get_bounds, OurHolland.com (2019);

Ranking

At start all entries are still ranked 0, this way we keep track of all the entries that are in the bounds and still to be ranked. We start the ranking process and begin the ranking at rank 1. If an entry is being totally dominated, meaning the competitors values are all at least the same and at least one value is better than the values of the current entry we are evaluating, then this entry we are evaluating is skipped meaning it's rank remains being 0 for now. We move in this manner through all entries comparing them with all all other entries except itself in a double loop fashion. All their values are compared and once we find any entries that are not dominated by any other entry, they are ranked with the current rank 1. Keep in mind that multiple entries can get ranked with the same rank because they dominate all other entries but amongst each other are incomparable because of they have different strengths and weaknesses thus one doesn't dominate the other. Once ranking of all the best entries as 1 has been finished, we move on to the next rank in this case 2 and start the process all over again. However the loop only considers entries that have remained 'unranked', this means ranked 0, this has been the case for first iteration too however then all of the entries were evaluated because all were ranked 0. Now we will evaluate all except the ones already ranked, in this case all the entries except ones with rank 1. This process iterates as long as there entries that still remain, having rank 0. Once all entries are ranked the computation has been completed.
After the computation all entries that were out of the bound and had an empty rank, not 0, thus were left out of the computation get ranked $X + 1$. Where X denotes the amount of unique ranks that were generated in the last computation , this means that these items not in the bound are ranked 1 more than the last rank of the previous computation.

The chosen bounds in the parameters section have an influence on the result of the ranking. They decide which entries are left out of the computation because they are out of bounds. This means that when maximising a criterion a minimum is chosen of 4 and the entry has < 4 for that criterion of when minimising a criterion a maximum is chosen of 6 and an entry has > 6 for that value it's rank is set to empty before the computation starts and thus it is left out of the computation loop and only afterwards it is ranked the worst rank possible, the last rank of the computation + 1, together with other entries that were left out of the computation for this same exact reason.

There is also a second effect the bound has on the outcome. We can illustrate this by the following simple example where the user has uploaded a CSV file of a list of hotels where the hotels have among other attributes the price per night and the amount of stars they have. Given that the user has chosen to minimise price per night and maximize stars and this table of hotels lists a hotel A that is cheaper than an other hotel, let's assume B, however given A having only 4 stars versus the 5 stars of the competitor B, if no changes are made to the bounds by the user A and B would be in the same rank considering they are not comparable because one is cheaper while the other has more stars. Now if the the same query is executed with user inserted bounds telling our system to chose three stars as a maximum bound meaning that further maximisation of the stars would not have any extra value for the user this would cause A to be ranked better than B because its cheaper and has over three stars. For entries with star values under 3 nothing would change in terms of the order of ranking however the rank itself can change because in this case A and B moved from a common rank to two separate ranks possibly enlarging the total number of possible ranks.

Result

The result of the query is displayed in a graph where the first chosen criteria is set out, on the x-axis, against the second chosen criteria, on the y-axis. The graph shows all ranked items, their names, values for the two criteria, rank and place in the graph. This is done using the CanvasJS graph library. CanvasJS is an easy to use HTML5 and Javascript Charting library. It runs across most devices including phones and desktops without compromising on maintainability or functionality. In a nutshell I have chosen to make use of this software because of the following points.

- Very simple and intuitive API
- Looks elegant on page
- Works on all modern devices
- It is standalone, no other library needed

CanavasJS charting library logo;

4 Coding

The landing page of the website is the CSV upload form. After this there are another six processes namely parsing the values to the screen, processing the values and inserting them in our database, choosing two attributes for maximisation or minimisation and the bounds, computing the skyline queries and updating the database and lastly showing the result.

getImport

We begin with programming the CSV upload form. In the controller all we have to do is this.

```
return view(’import’);
```

This is it. There is no data that we need to pass to the view and no more logic involved. In the view we create a from, with method=”POST” to make the upload possible. We take as an input the user selected file.

```
input id=”csv_file” type=”file”
```

And a checkbox indicating whether the file contains a header row.

```
input type=”checkbox” name=”header” checked
```

parseImport

The next step is parsing the imported file and showing it to the user for approval. So we will have $data array of rows, where each row will have an array of columns. Now, we can represent it as a table and give the user a choice to approve and move to the next part. We will first get the file.

```
$path = $request->file(’csv_file’)->getRealPath();
```

If the checkbox was checked the header will be sliced separately. If not then the header will be an empty array.

```
if ($request->has(’header’)) {
    $all = array_map(’str_getcsv’, file($path));
    $header = array_slice($all, 0, 1);
    $data = array_slice($all, 1);
} else {
    $data = array_map(’str_getcsv’, file($path));
    $header = array();
}
```

Both the $data variable and $header irrespective of the checkbox will get JSON encoded and uploaded to the CsvData table in the database together with the file of the CSV file and the checkbox value.

```
$csv_data_file = CsvData::create([
    ’csv_filename’ => $request->file(’csv_file’)->
       getClientOriginalName(),
    ’csv_header’ => $request->has(’header’),
    ’csv_headerdata’ => json_encode($header),
    ’csv_data’ => json_encode($data)
]);

$csv_data = array_slice($data, 0, 10);
return view(’import_fields’, compact(’csv_data’, ’
   csv_data_file’));
```

After that the slicing of only first ten lines of the file is done because they suffice to show the user how the file was interpreted and whether that was the intended way, if it is not and the user has made a mistake in the formatting then the user can go back to correct it in the CSV file before submitting again.

```
$csv_data = array_slice($data, 0, 10);
```

In the view the user is shown the contents of the data and has the option to go further to import the CSV.

```
return view('import_fields', compact('csv_data', '
    csv_data_file'));
```

processImport

When processing the file and uploading it to our database table named Skyline we first drop this table then create it again with a name column and an auto incremented id column which is the primary key.

```
DB::statement('DROP␣TABLE␣skyline');

DB::statement('CREATE␣TABLE␣`u2485d4152_lara1`.`skyline`
    ␣(␣`name`␣VARCHAR(255)␣CHARACTER␣SET␣latin1␣COLLATE␣
    latin1_swedish_ci␣NOT␣NULL␣)␣ENGINE␣=␣InnoDB');

DB::statement('ALTER␣TABLE␣`skyline`␣ADD␣`id`␣INT␣NOT␣
    NULL␣AUTO_INCREMENT␣FIRST,␣ADD␣PRIMARY␣KEY␣(`id`)');
```

We then create the necessary amount of columns corresponding to the columns in the original CSV file. We give them arbitrary names 0, 1, 2 etc.

```
$i = count($csv_data[0])-2;
foreach ($csv_data[0] as $column){
    if($i < 0 ) break;
    $alter1 = 'ALTER␣TABLE␣`skyline`␣ADD␣`_';
    $alter2 = $alter1 . $i;
    $alter3 = $alter2 . '`␣DOUBLE␣NOT␣NULL␣AFTER␣`name`'
        ;
    $i--;
    DB::statement($alter3);
}
```

Then we insert all data row for row, column for column.

```
foreach ($csv_data as $row) {
    $skyline = new Skyline();
    $skyline->name = $row[0];

    $i = count($csv_data[0])-2;
```

```
        foreach ($csv_data[0] as $column){
            if($i < 0 ) break;
            $columname = '␣' . $i;
            $csvrow = $i + 1;
            $skyline->$columname = $row[$csvrow];
            $i--;

        }

        $skyline->save();
    }
```

Lastly we insert a new column for the ranks, that we have not computed yet.

```
        DB::statement('ALTER␣TABLE␣`skyline`␣ADD␣`rank`␣INT␣NOT␣
           NULL␣AFTER␣`name`␣'); //add the rank with default 0
```

getBounds

In getBounds we request the two user chosen criteria, and whether they should be maximised or minimised, from the previous view. We retrieve the minimum and maximum values from these two criteria.

```
        $minValue1 = Skyline::min($columncriteria1);
        $maxValue1 = Skyline::max($columncriteria1);
        $minValue2 = Skyline::min($columncriteria2);
        $maxValue2 = Skyline::max($columncriteria2);
```

And we pass these values together with the values from the previous view we just requested, described above, to the next view called get_bounds.

```
        return view('get_bounds',compact('header_data'), [ '
           file_id' => $file_id, 'criteria1' => $criteria1, '
           criteria2' => $criteria2, 'columncriteria1' =>
           $columncriteria1, 'columncriteria2' =>
           $columncriteria2, 'minmax1' => $minmax1, 'minmax2' =>
            $minmax2, 'maxValue1' => $maxValue1, 'maxValue2' =>
           $maxValue2, 'minValue1' => $minValue1, 'minValue2' =>
            $minValue2 ]);
```

In the get_bounds view we let the user chose to change the bounds or leave them untouched. By default the minimum value for either of the criteria is the actual minimum value from the entries of this criterion in the CSV file, and the same applies to the maximum value. The user can choose to alter them to remove entries which don't pass a certain minimum or maximum threshold or make a value less important after a certain threshold e.g. lowering the maximum value when maximising a criterion indicating that after a certain boundary line maximising would have no extra positive impact on the user's choice for an entry in the CSV.

computeSkyline

After selecting the criteria, maximising and minimising, and the bounds all that information is present in this function and we can start computing the ranks of entries through the skyline query. First we set the opposite min max and bounds to use for the query, e.g. if the criteria is to be maximised, '>' then the opposite is minimisation '<', and

Rank is made possible to be NULL. It was NOT NULL first so that all entries would be ranked 0 by default.

```
DB::statement('ALTER␣TABLE␣`skyline`␣CHANGE␣`rank`␣`rank`␣INT
    (11)␣NULL');
```

Then we remove the entries that didn't pass the threshold by making them rank NULL instead of 0. In pseudo code it looks like this

```
UPDATE skyline
    (
                              SELECT  * FROM skyline S
                              WHERE
                                        S.criteria1    </> (if to minimise
                                            then > and to maximise <)
                                            minumbound/maximumbound (if to
                                            minimise minbound and to minimse
                                            maxbound)
                                        OR
                                        S.criteria2    </> (if to minimise
                                            then > and to maximise <)
                                            minumbound/maximumbound (if to
                                            minimise minbound and to minimse
                                            maxbound)
    )
    set rank = NULL ;
```

Now we arrived at the actual skyline querying. The actual code is rather long because of dynamically changing the values of the criteria and bounds through PHP in the SQL statement, however the representation in pseudo code is more understandable.

The query without the bounds would look like this for maximisation without loss of generality, for minimisation only the < should be flipped with > and vice versa.

```
SELECT  * FROM skyline S
WHERE   NOT EXISTS(
        SELECT *
        FROM skyline S2
        WHERE    S2.criteria1 >= S.criteria1 AND S2.criteria2 >=
            S.criteria2
                AND ( S2.criteria1 > S.criteria1 OR S2.criteria2
                    > S.criteria2 )
        );
```

This means that our current entry, and we go over all entries, gets selected then and only then if there is no other entry with the same or better value both in criteria 1 and criteria 2 and a better value in criteria 1 or criteria 2.

Now when adding the bounds part we have already eliminated the entries with unacceptable values through the SQL statement above making them rank NULL so now we are going to only consider entries a not-null rank. We have to also make sure that if there are two entries where one is better then the other but they are both over the boundary, that the user specified, after which having a better value is not important to said user we don't rank one above the other. When we add this part the query becomes like this, again for maximisation without loss of generality, for minimisation only the < should be flipped with > and vice versa as well as flipping the maxbound and the minbound and vice versa.

```
SELECT  * FROM skyline S
WHERE   NOT EXISTS(
        SELECT *
        FROM skyline S2
        WHERE   (S2.criteria1 >= S.criteria1 OR (S2.criteria1 >
            maxbound AND S.criteria1 > maxbound)) AND ( S2.
            criteria2 >= S.criteria2 OR (S2.criteria1 > maxbound
            AND S.criteria1 > maxbound))
                AND ((S2.criteria1 > S.criteria1 AND S.criteria1
                    < maxbound) OR (S2.criteria2 > S.criteria2
                    AND S.criteria2 < maxbound))
        );
```

This has the same principle as the last one only with the addition that there shouldn't be an entry with the same or better value in criteria 1 and criteria2 (or both current and other entries being above the maxbound, because maximisation has no added value above the maxbound) and a better value in criteria 1 (but current entry still under the bound, because maximisation still makes sense for the user under the maxbound) or in criteria 2 (but current entry still under the maxbound).

downloadFile

We give the user the possibility to download back the csv file with the computed ranks attached to the entries, in the same CSV file format on the result page. First the filename is set as skyline.csv

```
        $skyline_data = Skyline::all();

        $filename = 'skyline.csv';
        $handle = fopen($filename, 'w+');
```

After this using an array, first the columns are pushed in the array.

```
        $columns = Schema::getColumnListing('skyline');
```

```
$newarray = array();
foreach ($columns as $column){
    array_push($newarray, $column);
}
```

Then every row of each column and using fptucsv they are written to a CSV file which gets downloaded to the user's computer by calling this function through a button.

```
fputcsv($handle, $newarray);

foreach ($skyline_data as $row){
    $newarray = array();
    foreach ($columns as $column){
        array_push($newarray, $row[$column]);
    }

    fputcsv($handle, $newarray);
}

fclose($handle);

$headers = array(
    'Content-Type' => 'text/csv',
);

return Response::download($filename, 'skyline.csv',
   $headers);
```

5 Ranking Algorithm

Without loss of generality, let us in this part consider that all objectives need to be maximized. Note, that minimization goals can, in principle, be reformulated as maximization goals by taking the negative value (multiplying them with (-1)).

Let X denote a set of decision alternatives and f_1 ... f_m denote m decision criteria, that is $f_i(x)$ is the value of the i-th objective for some $x \in X$ and $i = 1$, ..., m.

We say point $x' \in X$ is said to be Pareto optimal for the problem if there is no other vector $x \in X$ such that for all f_1 ... f_m

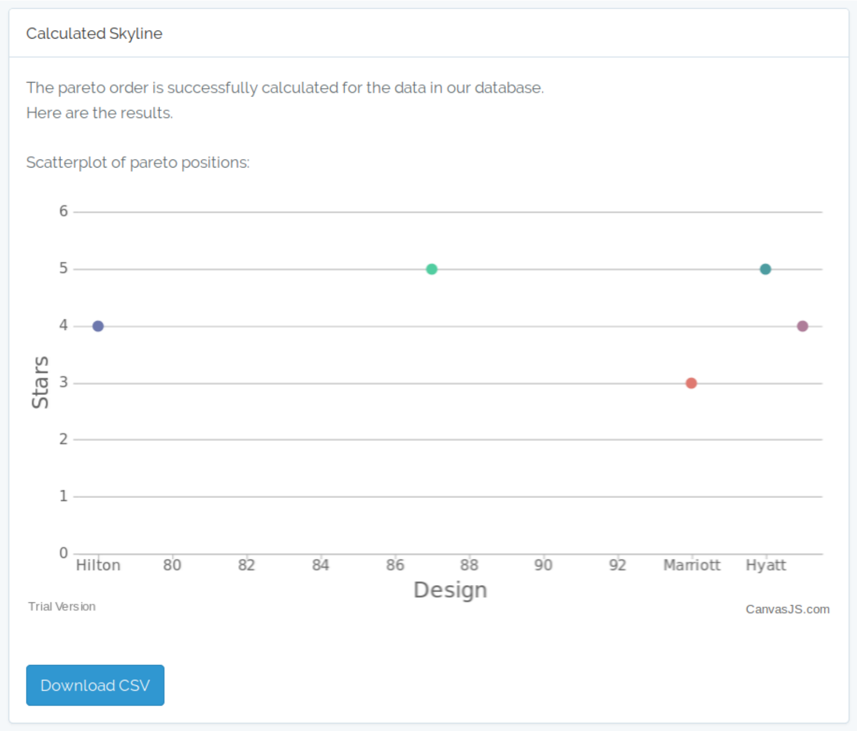
$$f_i(x) \succ f_i(x*)$$

6 Example

We import a CSV file with hotels and two criteria the user would like to maximise. The criteria are the overall design of the hotel and the amount of stars the hotel has.

```
Name,Design,Stars
Sheraton,87,5
Marriott,94,3
Hilton,78,4
Hyatt,96,5
Radisson,97,4
```

We choose to maximise both criteria and keep all the entries in the bounds. This graph represents the results.



Calculated Skyline

The pareto order is successfully calculated for the data in our database.
Here are the results.

Scatterplot of pareto positions:

compute_skyline, OurHolland.com (2019);

Hilton (blue) is ranked last as 3 because both it's design and stars are the worse than any of the other candidates. Marriott (orange) and Sheraton (green) rank as 2 because they are worse both in design and stars than the Radisson (purple) and Hyatt (dark green) however non-comparable to each other since one has a better design but worse stars and vice-versa. In the same way Radisson and Hyatt are also non-comparable to each other but better than any of the other entries thus they rank both as 1.

If we however would set the maximum bound for the Design on let's assume 86 then the ranking would look like this. The Sheraton and Hyatt would be ranked 1 because having design 87 is more then enough for the customer according to the maximum bound and in terms of stars these two are not inferior to each other, while the Raddisson would be ranked 2, because of the 4 stars but it being above the maxbound, the Marriott is still above the maxbound but has only 3 stars but is not inferior to the Hilton which has only a Design of 78 which is under the maxbound where that of the Marriott really is better for the customer but its stars are less than that of the Hilton which ranks them both at rank 3.

7 Conclusion

Yes, we can use relational databases to support skyline queries for a system that ranks solutions on multiple criteria. This is proven by the system implemented for this thesis which uses SQL, PHP and calculates the Pareto dominance with minimum and maximum bounds of an integer/float CSV file with multiple criteria using two of any the provided criteria in the CSV.

8 Further Research

With some extra research and programming we could find out how we could change the query and system overall to accommodate for more than two-dimensional criteria Pareto dominance calculations. A possible solution would be to let user pick for all criteria whether they are good values (maximisation) or bad ones (minimisation) then we could ask to select criteria up to the amount of criteria there are and do the querying not over two criteria but over all of the selected ones using a loop. In the graph we wouldn't set out one criteria against the other but the entries on the x-axis and their ranks on the y-axis.
We could change the creation of the table Skyline to making a new table for every request. Otherwise like it is now it is not possible for two user to simultaneously do calculations however this was done this way because not much traffic was expected. Not much programming would be needed to achieve this.
We could look into how we could program the application in such a way that it becomes possible to give a different certain maximum set using different methods, eg.

- all the best starting from ranking 1 going up
- one best solution, one median solution one low ranked solution
- one from each rank

We can also find out if we can alter the query (not use a nested SQL) to achieve a faster computation time and estimate/test them all to find out which method is the fastest? As well as find out what is the largest amount of entries we can process with the most used skyline query algorithms in under five minutes?

9 Relational Algebra

To further expand on the topic of involving all criteria into the decision making process as described in the Further Research section we will discuss how to achieve this using Relational Algebra.

The division is a binary operation that is written as R ÷ S. Division is not implemented directly in SQL. The result consists of the items in R that have a relation with all items in S.

We can use this to describe finding the items in our CSV file that the skyline query should find when incorporating all criteria into our search.

Let's assume we export our CSV file to the database by making an entry for each criteria for all items in a table called 'Solution'. This would look like this, however keep in mind that in database environment we need an extra column for a key (e.g incremented int).

solution	oncriteria	value
A	1	100
A	2	200
B	1	105
B	2	205
C	1	90
C	2	310

Now we could construct another table called 'Dominated' in our database that would say which items are dominated, by which other items and on which criteria.

A double for loop would compare all the items to all other items and if on the the same 'oncriteria' the 'value' of the other item is larger (when maximising) or lower (when minimising) that solution gets saved as well as by which other solution it is dominated and on which criteria. In this case it would look like this.

dominated	by	oncriteria
A	B	1
A	B	2
A	C	2
B	C	2
C	A	1
C	B	1

We have acquired all the data and tables we need. All we have to do now is choose which criteria we are incorporating in the search.

In the case of all criteria we use: (π oncriteria (Solution)) otherwise we can narrow it down using σ

To find if entry 'A' is being dominated by entry 'B' (on all criteria) we do as follows.

(π oncriteria σ dominated = 'A' $\wedge$ by = 'B' (Dominated)) $\div$ (π oncriteria (Solution))

To find all solutions that are dominated by any other for all criteria we can use a double loop. Pseudo code is as follows.

```
for( unsigned int i = 0; i < letters.size(); i++)
    for( unsigned int j = 0; j < letters.size() && i !=
        j; j++){

        ( pi oncriteria sigma dominated = 'i' ^ by = 'j'
            (Dominated)) / ( pi oncriteria (Solution))
    }
```

We could skip those solutions that are being dominated to find the ones that are in rank 1 in similar fashion as the original code we could go up incremented the rank until all entries are ranked.

Because in skyline queries an item is dominated if it is dominated on at least one criterion and is the same on all others we would have to change the relational algebra accordingly by introducing a third table or third column in Dominated with the name 'Same' for values that are the same.

dominated	same	by	oncriteria
	A	B	3

However this would make the the process very cumbersome so it is not described here but is certainly possible.

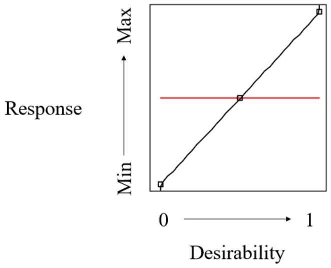
10 Related Work

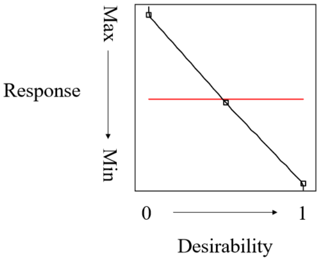
Closely related to this topic of Pareto dominance is desirability of an outcome, in this case called a response, of multiple decision makers and trying to build a model pleasing these decision makers.

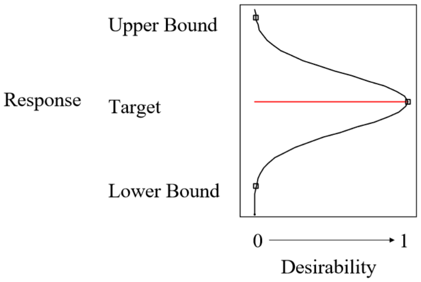
The problem in multiple response optimization is to find settings for multiple variables that calculate a desirable output for one or more decision makers. Each response, negative or positive has its distinct predictive model which can be used for optimization. When optimizing for one of the models this will in turn worsen the model for another response.

Desirability has been used a criterion for response optimization by Harrington (1965) and popularized by Derringer and Suich (1980).
The first step in defining a desirability function is to assign values to the response that reflect their desirability.

For every response, we define a function that has values between 0 and 1. 0 indicating no desire, and 1 indicating maximum desire. Then depending on the situation we maximise or minimise the response or try to match a target considering response (Min to Max) and desirability (0 to 1). Respectively for maximisation, minimisation and matching a target the charts look as follows.







11 Sources

Borzsony, Stephan, Donald Kossmann, and Konrad Stocker. ”The skyline operator.” Proceedings 17th international conference on data engineering. IEEE, 2001.

Skyline Queries by: Victor Mier 30.11.2010 Seminar Location-based Services (19562) Prof: Dr. Agnès Voisard

Multiple Optimization Using the JMP Statistical Software Kodak Research Conference, North Haven Group. May 9, 2005.

https://www.sharcnet.ca/Software/Ansys/16.2.3/en-us/help/wb_dx/dxBEMtemp10.html

https://laravel.com/docs/5.8