Visualization of the Optics Algorithm

Gregor Redinger[*] Markus Hunner[†]
01163940 01503441

VIS 2017 - Universitt Wien

ABSTRACT

In our Project we have the goal to provide a visualization for the OPTICS Clustering Algorithm. There hardly exist in-depth visualizations of this algorithm and we developed a online tool to fill this gap. In this paper we will give you a deep insight in our solution. In a first step we give an introduction to our visualization approach. Then we will discuss related work and introduce the different parts of our visualization. Then we discuss the software stack we used for our application and which challenges and problems we encountered during the development. After this, we will look at concrete use cases for our visualization, take a look at the performance and present the results of a evaluation in form of a field study. At last we will discuss the strengths and weaknesses of our approach and take a closer look at the lessons we learned from our project.

Index Terms: Human-centered computing—Visualization—Visualization techniques—TODO; Human-centered computing—Visualization—Visualization design and evaluation methods

1 Introduction

We developed a visualization for the OPTICS Clustering Algorithm (Ordering Points To Identify the Clustering Structure) [1]. This algorithm calculates a distance for each point in the dataset and uses this value to order the points in a meaningful way:

$$\text{dist}_{\varepsilon,MinPts} = \begin{cases} \text{UNDEFINED} & \text{if } |N_\varepsilon(p)| < MinPts \\ MinPts\text{-th smallest distance to } N_\varepsilon(p) & \text{otherwise} \end{cases} \quad (1)$$

The OPTICS algorithm is a prime candidate for the use of visualization techniques as its output of a ordered object list can be shown in a histogram-like plot (Reachability-Plot), that has to be interpreted by the viewer. "Valleys" in the plot represent clusters and as OPTICS is a hierarchical clustering algorithm "valleys" inside a "valley" can be interpreted as subclusters.

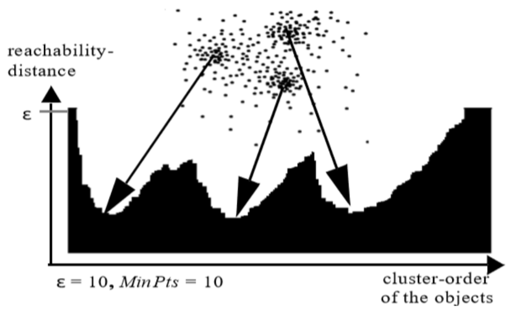


Figure 1: A example of the OPTICS output arranged in a Reachability-Plot in [1]

[*]e-mail: gregorredinger@gmail.com
[†]e-mail: hunnermarkus@gmail.com

Our Visualization not only helps in interpreting this Reachability-Plot, but also provides the functionality of picking a cutoff value for parameter ε, that we called ε', with this cutoff it is possible to interpret one result of the OPTICS algorithm like several results of the related DBSCAN clustering algorithm. Thus our visualization provides a parameter space exploration for all $\varepsilon' < \varepsilon$.

Users Therefore our visualization enables users without prior knowledge of the OPTICS algorithm and its unusual result format to easily interpret the ordered result structure as cluster assignments. Additionally it allows the user to explore the parameter space of the εparameter in an intuitive way, without the need to educate the user on the algorithmic details of OPTICS and why introducing a cutoff value for the calculated distance measures corresponds to the DBSCAN clustering algorithm.

Data Our Prototype provides three predefined datasets with the option to upload your own dataset in json format. The default dataset is a mockup dataset used to visualize the advantages of hierarchical clustering algorithms compared to non-hierarchical ones. It features several cluster arrangements, which need an hierarchical clustering solution to get right. With this dataset it's easy to explain how hierarchical clustering works and how the results have to be interpreted. As this is a dataset specifically designed to show off hierarchical clustering, it's easy to achieve perfect results.

To demonstrate the benefits of our visualization approach in a more realistic setting, we included two different 2D-Interpretations of Fisher's Iris flower data set [2]. The dataset features three clusters corresponding to the species "Iris setosa", "Iris virginica" and "Iris versicolor". While Iris setosa can be easily distinguished from the two other species, the clusters of Iris virginica and Iris versicolor are interwoven. This usually leads to results where those two are interpreted as one big cluster. Although OPTICS is unable to produce perfect or good results for this dataset, our visualization approach allows a user to easily identify that the "big" cluster holds more information than the OPTICS result suggests. By interactively exploring the parameter space for ε, the user is able to see that strict interpretations of what a cluster should be, lead to a dense area in the "big" cluster. This dense area usually corresponds to Iris versicolor or Iris virginica. With enough patience it is even possible to find a value ε' in the parameter space, where both species can be distinguished with the disadvantage of interpreting many points as noise. While those clustering results are still far from perfect, our visualization allows the user to assume that the result may be unsatisfactory and even gives hints what subspace or subset of the data should be examined with other categorization techniques.

2 Related work

Our Solution iterates on a proposed visualization of the output of the OPTICS algorithm in [1] when the algorithm was introduced: The Reachability-Plot (See 3.4). Using a cutoff ε' to select certain clustering assignments in the parameter space of ε- like we do - is also mentioned in [1] and [5]. Combining and linking a representation of the Reachability-Plot with a scatterplot can be seen implemented in the video [4].

A case study of hierarchical clustering visualization can be found in chapter 15.5 in [3]. While it discusses a much more advanced approach using a different clustering algorithm, our prototype and the discussed application share some similarities. For example it also combines scatterplots, histograms and tables to show different aspects of a clustering result.

3 Approach

3.1 Configuration

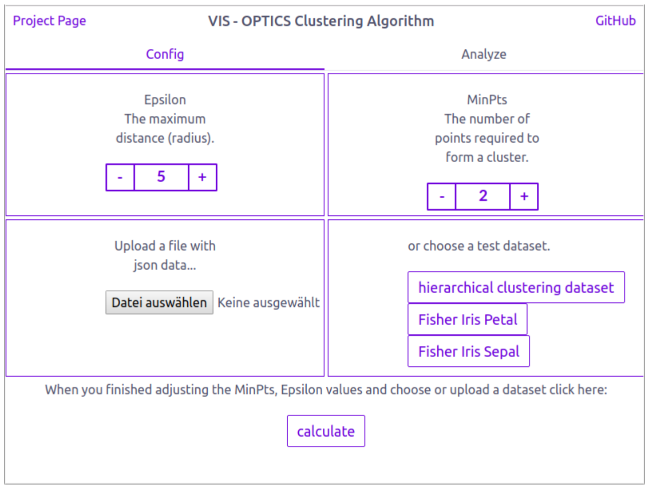

Project Page
VIS - OPTICS Clustering Algorithm
GitHub
Config
Analyze
Epsilon
The maximum
distance (radius).
- 5 +
MinPts
The number of
points required to
form a cluster.
- 2 +
Upload a file with
json data...
Datei auswählen Keine ausgewählt
or choose a test dataset.
hierarchical clustering dataset
Fisher Iris Petal
Fisher Iris Sepal
When you finished adjusting the MinPts, Epsilon values and choose or upload a dataset click here:
calculate

Figure 2: The config dashboard

We decided to split our application into two main sections. A configuration section, where the user can adjust the algorithm values and choose a dataset and a analyze section where the user can explore the data. This approach seems useful, because it prevents the site from getting to bloated with features and provides a clear interface to our users, because there are less elements on the screen.

The configuration section allows the user to adjust the minPts and epsilon values. We design the input field in a way, a user can increment or decrement the values by one after every click. So we tried to make it clear to the users, that it's better to adjust this parameters carefully and not to increment the epsilon value for example from 2 to 3000 at once. After we get some feedback from one of our professors, we also added a short explanation text to these parameters. At first we believed it would be enough to just write minPts and epsilon, because we would give every user a short explanation what the optics algorithm is about, before they use our application. After some test with colleagues we find out, that even if they get a introduction, they often forget the meaning of these parameters. So we implemented the text and during our final evaluations we noticed, that it was significantly easier for our users to understand the meaning of this parameters.

The configuration section also offers the option to upload a own dataset or pick a predefined one. We offer an hierarchical cluster and two versions of fishers iris dataset. We implemented a preview slider, to give our users a hint about the structure and content of this datasets. If the user click on one of the test-dataset buttons, the slider shows up on the left side of the screen like in Figure 3.

After all parameters are set, the user must click on calculate to start the calculation.

3.2 Table view

In the upper right corner we show the ordered list, which is the output of the OPTICS algorithm. All visualizations we implemented are based on this single ordered list of tuples (Point, Distance). See 6.3.2 Data Basis for a discussion of this constrain. In a second step our implementation assigns colors to presumed clusters based on their

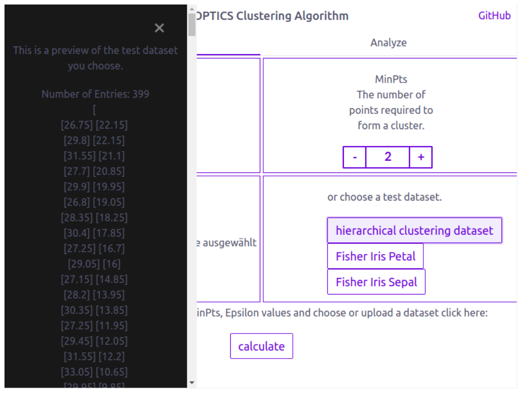
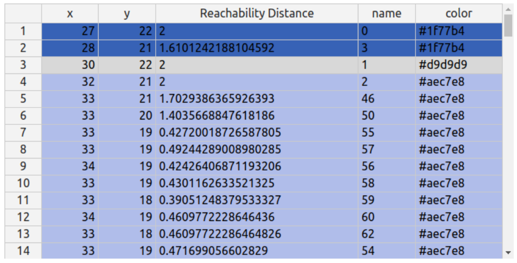

This is a preview of the test dataset
you choose.
Number of Entries: 399
[
[26.75] [22.15]
[29.8] [22.15]
[31.55] [21.1]
[27.7] [20.85]
[29.9] [19.95]
[26.8] [19.05]
[28.35] [18.25]
[30.4] [17.85]
[27.25] [16.7]
[29.05] [16]
[27.15] [14.85]
[28.2] [13.95]
[30.35] [13.85]
[27.25] [11.95]
[29.45] [12.05]
[31.55] [12.2]
[33.05] [10.65]
OPTICS Clustering Algorithm
GitHub
Analyze
MinPts
The number of
points required to
form a cluster.
- 2 +
ausgewählt
or choose a test dataset.
hierarchical clustering dataset
Fisher Iris Petal
Fisher Iris Sepal
inPts, Epsilon values and choose or upload a dataset click here:
calculate

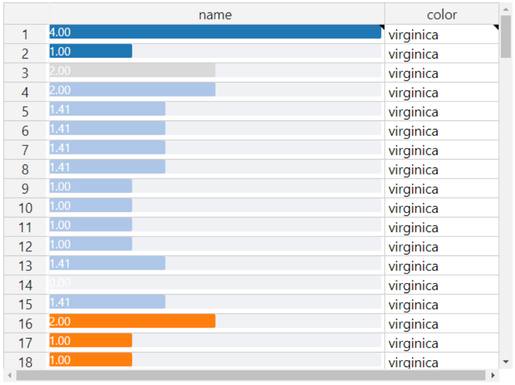
Figure 3: The test-dataset preview slidein

reachability distance. We visualized this color assignment with a bar inside the table representing the reachability distance in relation to εcolored accordingly. This bar provides a visual connection to the Reachbility-Plot discussed in 3.4 Reachability-Plot As a table is usually a textual representation of data, it is a suboptimal visualization. Nevertheless we wanted to show the algorithm results in way, that makes the connection between the input list of datapoints and the ordered output list of datapoints obvious.



	x	y	Reachability Distance	name	color
1	27	22	2	0	#1f77b4
2	28	21	1.6101242188104592	3	#1f77b4
3	30	22	2	1	#d9d9d9
4	32	21	2	2	#aec7e8
5	33	21	1.7029386365926393	46	#aec7e8
6	33	20	1.4035668847618186	50	#aec7e8
7	33	19	0.42720018726587805	55	#aec7e8
8	33	19	0.49244289008980285	57	#aec7e8
9	34	19	0.42426406871193206	56	#aec7e8
10	33	19	0.4301162633521325	58	#aec7e8
11	33	18	0.39051248379533327	59	#aec7e8
12	34	19	0.4609772228646436	60	#aec7e8
13	33	18	0.46097722286464826	62	#aec7e8
14	33	19	0.471699056602829	54	#aec7e8

Figure 4: A first iteration for the table view



	name	color
1	4.00	virginica
2	1.00	virginica
3	2.00	virginica
4	2.00	virginica
5	1.41	virginica
6	1.41	virginica
7	1.41	virginica
8	1.41	virginica
9	1.00	virginica
10	1.00	virginica
11	1.00	virginica
12	1.00	virginica
13	1.41	virginica
14	0.00	virginica
15	1.41	virginica
16	2.00	virginica
17	1.00	virginica
18	1.00	virginica

Figure 5: the table view now

3.3 Scatterplot

The scatterplot in the upper right of our prototype is a classical 2-dimensional representation of the input data. This form of visualization is one of the most common used in implementations of clustering algorithms and usually what users tend to expect, when confronted with unsupervised learning algorithms. Although it isn't a visualization that represents a result output of OPTICS directly, we decided to include it in our prototype as many potential users should be familiar with it. If a user is unable to interpret the Reachability-Plot, this view alone shows the clustering results by coloring the data points in accordance with Reachability-Plot. Highlighting points with the mouse courser highlights the corresponding point in the Reachability-Plot. This can help in understanding the connection between both visualizations.

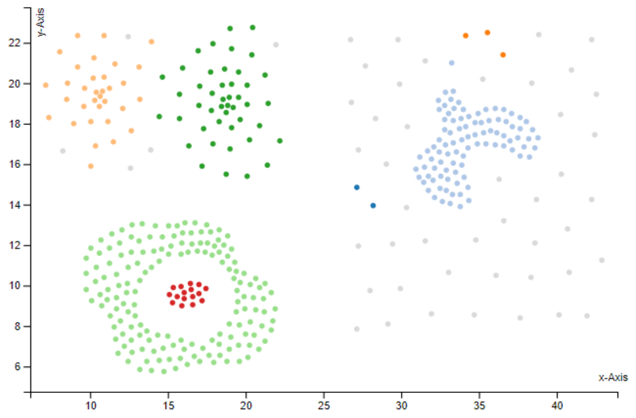


Figure 6: The scatterplot

3.4 Reachability-Plot

The idea of a Reachability-Plot is a central aspect of the OPTICS and was first mentioned in the corresponding paper [1]. The Reachability-Plot is in alignment with the ordering calculated by the algorithm. It provides a kind of birds-eye-view of the whole ordered structure and can therefore be seen as another way of visualizing the data our prototype already show via the table discussed in 3.2. By examining such a Reachability-Plot for "valleys" a user, familiar with OPTICS, can derive assumptions about the hierarchical structure of clusters in the data. Our visualization simplifies this approach by coloring the "valleys" for a given ε.

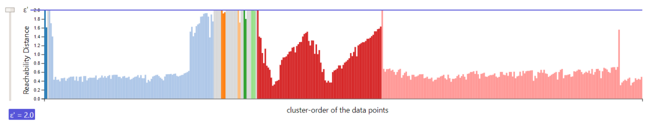


Figure 7: our interpretation of a Reachability-Plot

While this makes it easy to see the clustering result for a certain epsilon, it still doesn't visualize the hierarchical structure of "valleys" inside "valleys". For this we provide an input slider with which the user can explore the parameter space of εfor all $\varepsilon' < \varepsilon$. The coloring of clusters is changed accordingly. These clustering results can also be interpreted as the results of a run with the related clustering algorithm DBSCAN for selected εas discussed in [1]. Thus this input slider represents a form of local-to-global parameter space exploration described in [6] - by exploring several local results in one visualizations it is possible to derive a global structure of clusters over all ε'.

Our goal was to create a form of scented widget as discussed in [7]. Unfortunately technical complications (see 4.2.1) prohibited to put the input slider directly on the y-axis of the Reachability-Plot, which represents the distances. Therefore the slider is next to y-axis with a colored line representing the chosen ε'-cutoff.

3.5 Range-Query

OPTICS uses so called range queries in the calculation of reachbility distances. The radius for those range queries is ε. If

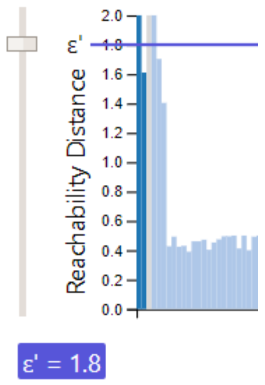


Figure 8: the input slider with the purple line indicating the current ε'-cutoff

there are at least *minPts* data points in this query the $MinPts - thsmallestdistancetoN_{\varepsilon}(p)$ is calculated otherwise the maximum distance εis assigned. To show an estimate of those range queries a data point can be highlighted in the scatter- or Reachability-Plot and circle corresponding to the range query is shown around the point in the scatterplot. This allows the user to check, if a point is regarded as noise because it didn't met the *minPts*-criteria.

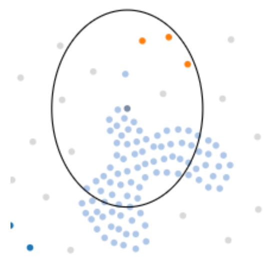


Figure 9: an estimate of the range query shown in the scatterplot.

3.6 Tooltips and highlighting

We used tooltips to provide further information about some parts of our application. Nearby each view in the analyze tab is a help button, by hovering over this button a tooltip pops up, that shows information on how to use the respective view. This is actually a feature we implemented later, because in the first place, we believed it wouldn't be necessary. After we run a few evaluations, we realized that even users that know the optics algorithm and other implementations of it, struggled to find all of the features we implement. So we decided to add this help buttons.

Another use case for the tooltips, was providing information we did not have enough space in the views. In the histogram, we haven't enough space to show information about each datapoint, so we decided to provide the respective information in a tooltip that shows up, when a user hovers over that point.

We use highlighting because we show large amount of data in our views and this makes it hard for a user to be sure which datapoint he have selected. So we provide a visual feedback. In the histogram view, we visualize this feedback with a color change to grey and in the scatterplot view a selected point gets a darker color tone.

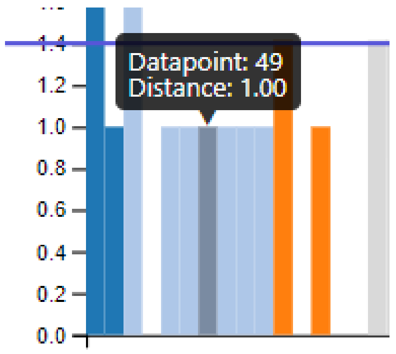


Figure 10: a tooltip providing the original index of the data point and the calculated distance

4 Implementation

4.1 Used tools and languages

4.1.1 Javascript

We used javascript as our programming language. This was a good choice because there are many libraries and tools for building web applications in this language. A downside of this approach is, that it's sometimes hard to choose a good library for a given problem, because there are so many of them and sometimes this libraries and tools don't play well together.

4.1.2 Webpack

We used a module bundler named webpack. So we were able to structure our code in a clean way by modularize it. Webpack takes care of stitching together all this modules (and their dependencies) into a single file in the correct order. A problem we encountered here, was the debugging of the bundled code, because the bundled and compiled code is a bit hard to read. After we find out how to use a source map (provides unbundled source code for debbuging) it was a bit better, but still not very comfortable.

4.1.3 Babel

We used babel as a preprocessor for our js code. This allowed us to use ES6 features(import/export, for of loops,...) and ensures that our code run in older browser too. The problem of the usage of babel in combination with webpack is that it caused a couple of problems with d3. A more detailed explanation of the problems we encountered follows in the next subsection.

4.1.4 npm

To manage our dependencies we used the node package manager. This was a good choice because it's much easier to manage all our dependencies in one place. Npm also takes care of updating our packages and warns us about deprecated packages.

4.1.5 JsDoc

To document our code we used jsdoc because it's very similar to javadoc, a tool we already gained a lot of experience with.

4.1.6 D3

Since D3 is the de-facto-standard for complex visualizations in javascript, we decided to use it as our tool of choice.

4.1.7 Handsontable

For the table view we used the library Handsontable. With this library it is possible to rendere html entities inside of table cells. We used this feature incorporate the reachability bars into the table view.

4.1.8 node + express

On the server side we used node in combination with express as server. This combination is very common and powerful for creating api's. The problem hereby is, that we actually don't do anything on the server side, except from starting the server and deliver our single page application one time on page load. A simpler approach would have been to just use the simply python server we used in A3. The only benefit we gained from taking this approach, was the usage of the powerful node package manager.

4.1.9 Jetbrains Webstorm

As IDE we used webstorm and it was a good choice, because it's especially designed for web development and very reliable and powerful. However we encountered a downside here, because the debugger don't work well with chrome or any other browser we tried. This makes the debugging of our javascript code not easy.

4.2 Challenges and Problems

4.2.1 d3 in combination with babel and webpack

We encountered an unresolved bug[1] regarding the d3-library and the usage of the javascript bundler webpack and the es6 to es5 compiler babel.

The d3 documentation states: "If you use Babel, Webpack, or another ES6-to-ES5 bundler, be aware that the value of d3.event changes during an event! An import of d3.event must be a live binding, so you may need to configure the bundler to import from D3s ES6 modules rather than from the generated UMD bundle; not all bundlers observe jsnext:main. Also beware of conflicts with the window.event global."

The Problem hereby is, that Babel treats imports (like d3.event) as values and not as live bindings. This makes it not possible to use d3's brushing when using a es6 to es5 compiler. Listener functions called via an event will throw nullPointerExceptions, if the reference to the d3.event was lost completly. In other cases the reference will point to another event object on random leading to TypeErrors inside the listeners.

First we take into consideration to forget about the use of webpack and babel in our code, but this would force us to restructure our project entirely. We decided that our project would benefit more, if we invest our resources in improving other parts of our application instead of restructuring huge parts of our application.

4.2.2 Dendrograms in d3

A dendrogram encodes clusters and there subclusters by visualizing "split-values" (ε'), at which a big cluster splits into two or more subclusters.

M2 featured several visualizations utilizing cluster dendrograms. This type of visualization would be a perfect fit for our use case, as it encodes the same information as the OPTICS-Reachability-Plot in a completely different way. Therefore it would be easy to derive the necessary data needed for a dendrogram from our reachability results as discussed in [5].

Unfortunately there exist no easy way to generate those with d3. The suggestion most often found, advises to use a visualization similar to what we called Graph- or Tree-View in M2. Unfortunately this means to lose the information encoded by the single axis of a dendrogram, the ε-Axis. While a tree view still perserves the information on cluster hierarchies, it is not possible to decide on a possible ε-cutoff with this form of visualization. Additionally we lose the possibility to link our ε'-Slider with the dendrogram. So we decided that we drop the idea of implementing a dendogram and focus our resources on other parts of the application.

[1]d3 Issue Tracker: https://github.com/d3/d3-brush/issues/9

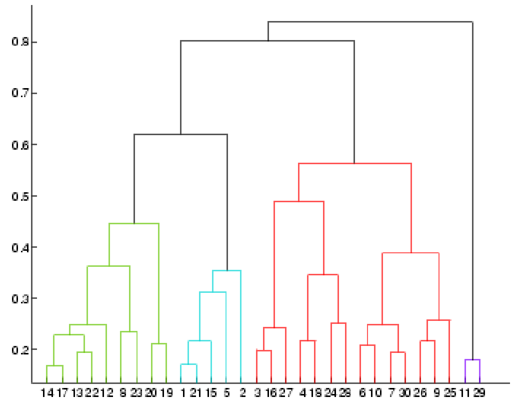


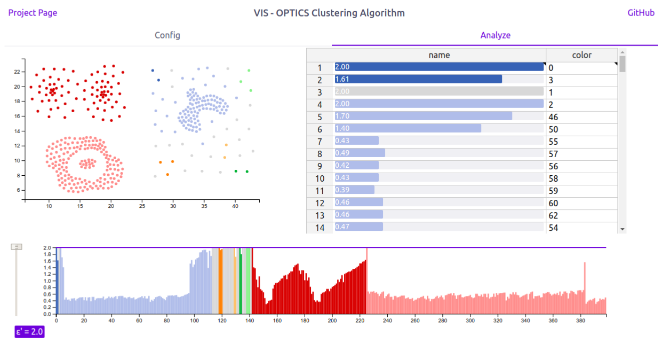
Figure 11: An example of a simple dendrogram

5 Results

5.1 Use Cases

5.1.1 Finding Customers with a high Income that lives in a specific street

Frank, the owner of a local grocery store wants to offer his customers the option of a 10 Minute delivery, to compete with Amazon Fresh, that started recently in his city. Because a 10 minute delivery is not possible in the whole city, he plan to restrict his offer to a specific street. He also thinks that only people with a relatively high income are willing to pay the delivery fee, so he want to limit his marketing to these kind of people. Thankfully he had a list with the income and the residence of all people of his city as json. The problem is to evaluate the list by himself would take way too much time, so he decided to use our application instead.



	name	color
1	2.00	0
2	1.61	3
3	2.00	1
4	2.00	2
5	1.70	46
6	1.40	50
7	0.43	55
8	0.49	57
9	0.42	56
10	0.43	58
11	0.39	59
12	0.46	60
13	0.46	62
14	0.47	54

Figure 12: The analyze Dashboard

At first he imports his list Data in our application and adjust the epsilon and minpts parameters. After analyzing the scatterplot (left top), he quickly see that the customers with high income are in a blue cluster in the top-right of the scatterplot. Now he searches in the spreadsheet view (right-top) after all entries with the street, where he plan to offer his delivery service. Because all clusters are colored in the table view, he can complete his search in no time.

5.1.2 Offering tutoring classes

Mr Fisher is the principal of a high school and he plan to offer some tutoring classes to his pupils, but because his budget is limited, he is only able to build a limited number of that classes. So he decided to only offer them in disciplines where many pupils have problems. He had a list of his pupils and their grades and analyzes the list with our Application. He uploads the json file and click calculate. Then he switches to the analyze tab. He doesn't need help during this workflow, because the concept "first config then analyze" seems very natural and logical to him. He adjust the epsilon value with the histogram view and examine the clusters in the scatterplot. He decided to examine the clusters even further and switch to the spreadsheet. The distance column is colored in the cluster colors so he is able to find the list of pupils, belonging to a specific cluster really fast.

Now he knows which pupils had problems in specific disciplines and where he can offer tutoring classes with the highest efficiency.

5.2 System Performance

Or application is browser based and therefore the performance cannot be compared to native os applications in c or c++. Especially for large datasets our application won't perform as well, as as native solutions, but for mid-sized datasets our application will perform quite well. A performance evaluation for a dataset containing 400 elements:

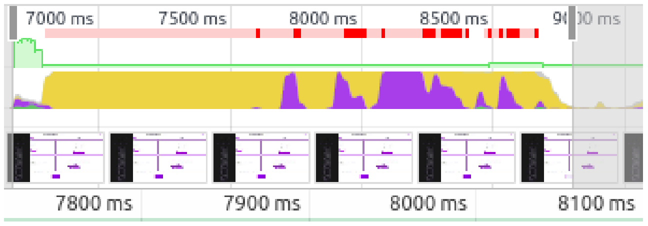
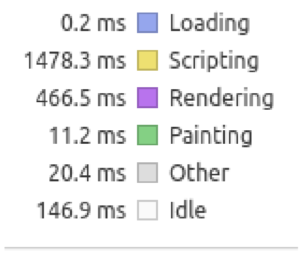


Figure 13: The image shows the system utilization from clicking the calculate button to finishing the calculation and rendering. The yellow color shows scripting and the violet color rendering tasks. (created with chrome dev tools)



0.2 ms	🟦	Loading
1478.3 ms	🟨	Scripting
466.5 ms	🟪	Rendering
11.2 ms	🟩	Painting
20.4 ms	⬜	Other
146.9 ms	⬜	Idle

Figure 14: An image of the calculation and drawing time of the different tasks. As expected the calculations of optics (scripting) consumes much more time than rendering the data to the screen. (created with chrome dev tools)

5.3 Evaluation

Our evaluation goal was to test our application in a realistic setting, but it should also be possible to perform precise observations. So we decided to use the methodology of a field experiment. Our test group consists of seven people, all of them were students in computer science. Since the topic of clustering algorithms, that we covered in our application, is not as well known as other topics in computer science, we decided to give the participants of our study a short introduction. In this introduction we covered what clustering algorithms are and their use cases. The participants were asked to perform the following tasks:

- Configure the algorithm and start the calculation.
- Switch to the analyze dashboard.

- Explore the dashboard and tell us what you think that the views might represent.

- Change the epsilon value in the histogram.

- Tell us why the colors are changing.

Thanks to our introduction to optics and the info-text nearby the epsilon and minPts, the participants had no problems to adjust the parameters and understand what they mean. A problem occurred by choosing a test dataset. By clicking on one of the test dataset buttons, a slider with information about the selected dataset pops up on the left side of the screen. It was pointed out in the slider that the dataset, the user clicked on, was selected. All off the participants understand what the pop up shows, but five of them mentioned, that it isn't clear how to choose a dataset, because there is no sign on the button itself, that the dataset was chosen. After asking the users if they don't read the text in the preview, which says the dataset was chosen, three of them responded they don't read the preview text and two of them where not sure if it worked, because the button of the selected dataset don't change his appearance.

The next issue we encountered refers to another design problem. After the user clicked on the calculate button, it is required to click on the analyze tab to switch to the analyze dashboard. When the participants finished the configuration and clicked on the calculate button, we instruct them to switch to the analyze dashboard. All of them responded, that they already click on the calculate button, but the application seems not to be finished with loading the new page. None of our users understands, that they had to click on the tab analyze to switch to the other dashboard, they expected to get redirected. This is actually obvious, but when we create the application we design it that way and after doing this task a few times, we don't even think about the fact, that they might be a more natural solution for this task, like getting redirected after clicking on calculate.

The majority of the participants mentioned, that there seems to be a connection throughout all the views, because of the same color scheme in all views. Six of the participants understand without further help that if elements have the same color, they belong to the same cluster. All of the participants recognized, that there is a linking between the elements they are hovering in the histogram and the elements in the scatterplot. Two of the participants understands that the circle that shows up by hovering in the scatterplot refers to the value epsilon. The table showing the reachability distance as a bar chart and the flower type of the fisher datasets was also quite clear for most of the users.

The task of changing the epsilon value was also no problem for our participants and except of one user, all of them understand that by changing the epsilon value, the clusters are reevaluated. In summary, we get a really positive feedback on our application. The participants liked the clear structure and the various ways to explore the datasets. The Evaluation provided us also with a lot of interesting ideas and show us the two major weaknesses of our implementation. The first of them is, that we forget to use standard feedback patterns in parts of our Application. This refers to the dataset buttons that should provide a visual feedback, which indicates that they were selected by the user. It's also very interesting that we totally omit the auto redirection after clicking on the calculate button, but this shows how important feedback is and how easy it is to overlook better interaction approaches. The second major weakness of our application is a lack of guidance. Especially in the analyze dashboard it would be a great improvement to provide some sort of tutorial, that shows how to interact with the views and which methods are available to explore the data in more detail.

6 DISCUSSION

6.1 Strengths of our approach and implementation

Iteration on OPTICS Our visualization in a iteration on ideas and concepts already connected with OPTICS. As can be seen in 1, the idea of connecting a scatterplot like view to the information encoded in the Reachability-Plot was already present in the initial publication of the algorithm. We iterated on the concept of the Reachability-Plot by introducing color and interactive exploration. This allows the users to interpret our visualization without prior knowledge of [1].

Usability During the development of our application we increased the usability continuously. Thanks to the participants of our evaluation we get a lot of feedback and now, at the end of this iterative process, our application provides a smooth user experience. There is some potential for improvements, but under consideration of our limited time resources for this task, we implement a lot of features:

- We implement a preview of the selected dataset in the config tab, so a user get more insight in the respective data.

- We implement tutorial buttons for each view, which provide information about the respective view and what a user can do there.

- We redesign our spreadsheet view entirely, to provide more valuable information.

- We improve the readability of the axis

- We redesign our config tab

Open Development We put a link to our github repository as well as a link to our project page in the header of our application. We want to encourage our users to look at the theoretical background of our implementation on our project page and examine and fork our code on github. We focused on a detailed documentation and we provide an even more detailed readme to make it as easy as possible to work with our code.

6.2 Weaknesses

Parameter Space Exploration While our input slider for ε' provides a form a parameter space exploration for εour visualization does not include parameter space exploration for *minPts*. Possible ideas to implement this could feature a side by side view of several different Reachability-Plots. As those could feature a vastly different ordering for the given input data, a form of mapping the same data points to each other in all of those plots would be necessary.

Table view The table view visualizes nearly no insightful information, except the connection between the output of OPTICS and the Reachability-Plot. As the table view provides a kind of "zoomed in" view of the calculated results, a zoom and brushing functionality for the Reachability-Plot could probably replace the whole table view. Unfortunately technical difficulties prohibited the implementation of such a feature (See 4.2.1)

Linking of visualizations Currently there exist no strong links between our visualizations. The color assignments are present in all visualizations. As is the possibility to highlight a point in the scatterplot or the Reachability-Plot and see the corresponding data point in both of those. Ideally it should be possible to come up with a stronger connection between the Reachability-Plot and the Scatterplot. For example some visualizations draw a line from a "valley" in the Reachability-Plot to the corresponding cluster of points in the scatterplot. For a very small number of clusters this can still be interpreted, but doesn't scale well. Further Iterations on the concept should include a more interactive way to connect clusters in both plots without the usage of static connection lines.

6.3 Lessons we learned

6.3.1 Choose the right tools

One of the most important things we learned from our project, is that it is crucial to be not too biased to use a specific tool or handle things in a specific way. One of our professors in the course described this problematic point of view quite good: "If your favorite tool is a hammer, you tend to see every problem as a nail". In our project we used babel and webpack, because we already used them in other projects and were familiar with them. We forgot to check if this tools will work well for the problem we want to tackle. The thing we learned here is to look at a problem and then choose a tool to solve it and not the other way around.

6.3.2 Data basis

On several occasions we were advised against implementing OPTICS on our own and look for a already implemented solution to use. As OPTICS seems to be a very exotic clustering algorithm there hardly exist well written implementations that allow to access interim results or execution data. And most of the implementations in javascript seem to be student projects.

Many of our low-fi-prototypes relied on data not accessible by the OPTICS implementation we used. Like the exact data point traversal used during execution, which we wanted to visualize in what we called a Net-Graph.

Other interim results like the range queries performed by the algorithm for every point in the dataset could be easily saved during the execution to use them for a visualization. As of now, we need to "reengineer" the neighborhood query from the result output, which leads to inaccurate estimates.

Another group which worked with the OPTICS algorithm implemented their own version of the algorithm and were able to come up with much more visualizations, as there data basis after running the clustering algorithm consists of much more information than just an ordered array of data points.

Data used in a run of the optics algorithm that is inaccessible to us includes, the range queries and for which point it achieved to reach minPts, the dataset traversal, and real clustering assignments. All visualized clustering assignments are currently derived from the ordered result list, as intended for human users by [1]. Storing calculated cluster assignments would allow us to give a much more consists visualization by enabling us to track a certain cluster through the whole parameter space of ε.

There are only so much possible visualizations of an ordered array. This leads to the conclusion that a proper data basis is crucial for a good visualization. A point we realized far too late during the implementation and should have been thought of already in the stage of conceptualization and during the design of low-fidelity prototypes. As of now many of our low-fi prototypes rely on data we are not able to access in our high-fidelity prototype.

6.3.3 Leave well trodden paths

To choose the right visualization for a given problem is not an easy task. There are many things to consider. Who will use our application for what? What is the best way to show the data for the required tasks? How can i provide the user a smooth user experience? Confronted with so many decisions to make, we tend to look at the things other people do, to handle similar problems. There is nothing wrong with that, comparing our ideas with others can provide new insights to a problem and can help us to enhance our own ideas. The problem starts when we stop thinking about a problem by ourselves and just copying the things other people do, because everybody says this is the best approach. We believe it's crucial to think beyond this, to explore new ideas and even we fail a few times, we can learn from that mistakes and get insights we didn't have before.

7 Separation of Tasks

7.1 Markus Hunner

- Rework of table view
- literature research
- 50% of the Report

7.2 Gregor Redinger

- Design improvements in the config view
- Implementation of help buttons
- Dataset Preview
- Evaluations
- 50% of Report

References

[1] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. Optics: ordering points to identify the clustering structure. In *ACM Sigmod record*, vol. 28, pp. 49–60. ACM, 1999.

[2] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of human genetics*, 7(2):179–188, 1936.

[3] T. Munzner. *Visualization analysis and design*. CRC press, 2014.

[4] B. Özerdem. Optics clustering algorithm simulation. YouTube, December 2014.

[5] J. Sander, X. Qin, Z. Lu, N. Niu, and A. Kovarsky. Automatic extraction of clusters from hierarchical clustering representations. *Advances in knowledge discovery and data mining*, pp. 567–567, 2003.

[6] M. Sedlmair, C. Heinzl, S. Bruckner, H. Piringer, and T. Möller. Visual parameter space analysis: A conceptual framework. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2161–2170, 2014.

[7] W. Willett, J. Heer, and M. Agrawala. Scented widgets: Improving navigation cues with embedded visualizations. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 13:1129–1136, 2007.