Combining Degree of Interest Functions and Progressive Visualization

Marius Hogräfer, Dominik Moritz, Adam Perer, Hans-Jörg Schulz

Abstract

When visualizing large datasets, an important goal is to emphasize data that is relevant to the task at hand. A common way of achieving this is to compute the relevance of the data using degree of interest (DOI) functions, which apply a scenario-specific metric to quantify the data items according to their relevance to the users and their tasks. These DOI values can then be used to adjust the visual encoding through mechanisms like focus+context or information hiding. For datasets too large to be visualized at once, an alternative approach is to visualize it progressively in chunks, allowing analysts to reason about partial results and concluding their analysis much earlier than had they waited for all data. Combining the advantages of both approaches to tailor the visualization seems synergistic, yet, in practice turns out to be challenging, as DOI functions require the context of all data to produce useful values, requiring lengthy computations that break analysts' flow in progressive visualization. In this paper, we propose an approach for uniting DOI functions with progressive visualization. We first introduce a new model for quantifying the user interest in analysis scenarios where the data is only partially available, by computing the interest for available data and predicting it for the rest. We then propose regression trees for implementing this approach in practice and evaluate it in benchmarks. With DOI values now available for progressive visualization as well, our approach opens the door for tailoring the visualization of large datasets to the analysis task at interactive update rates.

Keywords: Progressive visualization, Degree-of-interest functions.

Index Terms: Human-centered computing—Visualization—Visualization application domains—Visual Analytics;

1 Introduction

Degree of Interest (DOI) functions [7] are an important method underpinning many approaches for interactive visual analysis. Based on scenario-specific metrics, DOI functions quantify the user interest for parts of the data, which then allows for tailoring the visualization to the analyst's interest. The general idea is to use the interest values to present interesting parts of data prominently, rather than having analysts waste time digging through data uninteresting to them. Well-known techniques enabled by DOI functions include focus+context views [10], interest-based labeling [1], and information hiding [4].

DOI functions share their goal of tailoring visualizations of large datasets with progressive visualization methods. In progressive visualization, the dataset is divided into subsets (so-called chunks), which are then visualized incrementally over time, rather than visualizing the entire dataset at once [23]. Progressive visualization can be tailored to analysts' interests by prioritizing data that are most relevant, i.e., visualizing them as early as possible, while showing less relevant data later or never. Benefits of the progressive approach are the ability of terminating otherwise long-running computations early, once all interesting data has been visualized [27], dynamically steering the computation to prioritize data from regions of interest [26], and ensuring the display of the most interesting data regardless of the available display space [21].

- *Marius Hogräfer and Hans-Jörg Schulz are with Aarhus University. E-mail: {mhograefer,hjchulz}@cs.au.dk.*
- *Dominik Moritz and Adam Perer are with Carnegie Mellon University. E-mail: {domoritz,adamperer}@cmu.edu.*

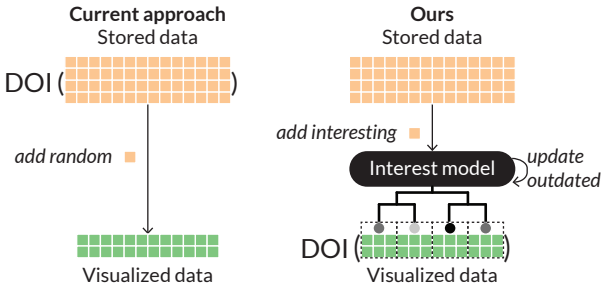

Figure 1: We reach interactive update rates for DOI computations on large datasets by progressively updating a user interest model when data relevance changes, rather than computing it on all data.

Thus, DOI functions and progressive visualization both aim to make the visual analysis of large datasets more manageable by taking the user interest into account: DOI functions allow tailoring the visualization to visually emphasize data items based on user interest, while progressive visualization leverages user interest to show interesting data as early as possible – realizing a "temporal emphasis" of sorts. This similarity has also been noted in prior work, for example, by Stolper et al., who mention a benefit of progressive visualization being that analysts can move "patterns of interest" to the front of the processing queue [23], or by Hellerstein et al. who propose a mechanism for dynamically reordering the data whenever analysts "change their definition of interesting" [12].

With so much overlap between DOI and progressive visualization, this paper investigates the idea of joining these two approaches. However, their combination leads to challenges caused by the disparity between what data is needed to compute a DOI function and what data is available in the progressive visualization. In non-progressive visualization, where the entire dataset is available, computing a DOI function is conceptually straightforward, but how does one do it when only having access to the data one chunk at a time? A naïve approach to compute a DOI function is to compute it only on every incoming chunk of data, but that clearly disregards all other data visualized so far. An alternative is computing a DOI function over all visualized data with every chunk, but that also quickly takes too long, as more and more data is visualized. Thus, while DOI functions could help increase the efficiency of progressive visualizations, these challenges need to be resolved first.

In this paper, we address these challenges by first proposing a new model for the user interest on progressive visualization, and we then show how to use it in practice. Specifically, we propose to move from exhaustive DOI computations to partial DOI predictions: Since we cannot efficiently compute the DOI function over all stored data, we instead compute it over a small subset of the data and train a predictor on its output. We then use this predictor to maintain valid interest values for the visualized data and to include new relevant data from the database in the visualization (see Figure 1). We show how the hierarchical structure of regression tree models are ideal for making these selections on tabular data, and demonstrate the feasibility of our approach in benchmarks.

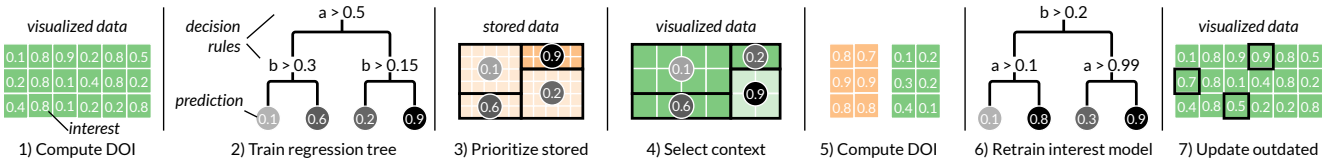

Figure 2: Demonstrative example of our approach: (1) Computing the DOI function over visualized data assigns an interest value to each data item. (2) Then, a regression tree is trained on that data, using the interest as labels. The trained model consists of a set of decision rules that divide the data by predicted interest value. (3) These decision rules can now be used to query the stored data, by appending them to the query that retrieves the next chunk. Appending rules leading to a high predicted interest steers the progression towards interesting data. (4) To supplement the steered sample, we enrich it with visualized data selected along "uninteresting" paths of the regression tree. (5) The DOI values are then computed for the selection from the two previous steps, and (6) a new regression tree is trained. (7) By comparing the predicted interest with the earlier regression tree, we can detect changes per path in the interest model and update the interest values efficiently.

2 Background

A central goal for useful visualization is to show the most relevant data to analysts at all times. To know what data is relevant, degree of interest (DOI) functions can be used that compute the current user interest for a data item relative to all other items in the dataset. User interests are complex and highly scenario-specific, and so are DOI functions computing them. The output of a DOI function is usually normalized to the range of $[0, 1]$, with 0 indicating an item is of lowest interest and 1 for highest. DOI functions can be complex, and are often a linear combination of different terms capturing specific facets of the user interest that make an item relevant to the analyst. For example, terms may measure a priori interests (e.g., a large item value) [7], but it can also depend on interest values of its surrounding items in the current visualization [1] or on the interaction state, such as the distance to the current focus node [11], the similarity to the active search terms [25], or the number of times analysts have already interacted with a data item in the past [9]. Thus, only some DOI terms can be pre-computed a priori; others can only be computed during the analysis. Additionally, while some DOI terms can be reused across scenarios, this is often complicated since they are highly scenario-specific, for example to aid the visual exploration in specific ontology tools [2].

Once the user interest is computed, the visualization can be tailored to these interest values to emphasize relevant parts and de-emphasizing irrelevant parts. On multivariate datasets, for example, interest values can drive focus+context visualizations across multiple views, making the most interest data stand out in each [3].

But DOI functions have a problem: keeping up to date with what data is relevant takes a long time on big datasets. This is because DOI functions are relative, meaning that they compute the user interest relative to all other data, requiring access of the entire dataset. While pre-computing the user interest would be a simple solution, real world interest values are constantly changing, caused by users selecting a different focus node [8], changing parameters/terms/weights of the DOI function [1], entering new search terms [25], or moving on to a different task based on their acquired knowledge [9]. In some cases, these dynamics can be so complex themselves that they become a subject of analysis in their own right [22]. What is challenging here is that every time the data relevance changes, we need to compute the DOI function over the entire dataset, and the bigger our dataset, the longer that computation takes. In other words, keeping up with relevance on large datasets by computing a DOI function can simply take too long for "fluid" interaction [5].

The open research challenge then is how to still tailor progressive visualizations of large datasets to the task relevance of the data. In response, we propose to compute the DOI on a data subset, using a model of the user interest to fill in for the rest of the dataset (see Figure 1), rather than computing a DOI function over all data every time the data relevance changes (which happens frequently). While no model is perfect and inherently uncertain, predictions are still a valid option, because DOI functions themselves are inherently only approximations of the "actual" subjective relevance to the analyst. Therefore, small deviations between predicted and computed user interests may be negligible for the visualization. More importantly, a model allows us to efficiently update the visualization incrementally, rather than requiring full replacements. In our approach, whenever the relevance of the data changes, a data item either becomes interesting, it becomes uninteresting, or it stays interesting. By maintaining a model of the user interest, we are able to identify these changes and, thereby, limit the data used for the actual DOI computation to keep update times low enough for fluid interaction.

As a result, the benefits of progressive visualization complement those of DOI functions. Incremental updates allow us to tailor visualizations of large datasets to the user interest without delay, because we can focus the DOI function on a small subset of data that needs it. That way, we enable the interest-based interactive progressive visualization of massive datasets.

3 A Progressive Approach to DOI Functions

Our approach divides the data into two parts: visualized items are most relevant to the task and are, as the name implies, shown in the visualization, and stored items, essentially all other items in the dataset. To ensure that the visualized data is relevant, we maintain a model of the user interest using regression trees. Regression trees recursively split the dataset along its attributes into two similarly relevant subsets. Each leaf node of a trained regression tree predicts an interest value of data matching a set of decision rules.

While there are many other (and in some cases more accurate) prediction methods, the hierarchical structure of regression trees makes them ideal for our purposes, because it allows us to query the data based on its predicted interest. That is, by traversing the tree from a leaf node (i.e., a predicted interest) to the root, we obtain a Boolean filter that describes items with a predicted interest. In other words, from a relatively small training sample [18] a regression tree enables us to efficiently divide the data – both visualized *and* stored – into high or low interest partitions, without actually computing the DOI function over all data. This extends upon our prior work [13], where we used trees to efficiently steer progressions towards unprocessed data subspaces similar to a selection in view space. Having a prediction of the user interest over the entire dataset then allows us to keep the visualization most relevant at all times. We detail this process below, aided by the demonstrative example in Figure 2.

Assumptions We focus on tailoring visualizations of a numerical, tabular dataset that is too large to fit into memory or on the screen and is, therefore, visualized progressively. We also assume that an appropriate DOI function exists for computing the user interest, and that the visualization does not contain any data, yet.

To be characterized as progressive according to the definition by Fekete and Primet [6], updates to the visualization need to happen within the human latency limits. In other words, there exists a maximum number of items c_{max} that the DOI function can process with each step, while ensuring that the visualization remains interactive.

Overview The general way in which our approach works is as follows (see Figure 2). When the visualization is empty at first, we

draw a sample of size c_{max} from the visualized dataset, and compute the DOI function over it, producing a numeric interest value for each sampled item. We then train a regression tree on the sample, using the computed interest as labels. With this regression tree, we can now efficiently predict the interest of the remaining data.

We use this to progressively update the visualization. For example, when selecting the next chunk of data, we use the tree to prioritize interesting data. To do so, we generate the Boolean filter for the leaf node with the greatest interest by combining the decision rules that lead to it in a conjunction. We append this filter to our database query that retrieves the next chunk, thus steering the progression. In return, we get a set of data items that the model predicts to be of high interest. Before computing its actual interest, however, it is important that we supplement this sample to avoid biasing the DOI computation. We can again use the tree to achieve this by selecting visualized data along all other leaf nodes except for the one we used in the previous step. We then compute the DOI function on the combination of interesting and uninteresting data and train a new regression tree on the output. Before updating the visualization, we need to ensure that all interest values for data so far in the visualization are still consistent with that new model. We do so by comparing the predicted interests for items from all leaf nodes of the new regression tree with their previous interest. Whenever the two differ significantly for a leaf node, we predict the for all items of that node. We repeat this process of retrieving new items and adding it to the visualization until all data is visualized. Below, we elaborate on the steps of our approach.

Training the first regression tree Since the visualization at first does not contain any data to compute an interest on, we begin by selecting a random set of c_{max} stored items in order to get a broad first overview of the data. Then we compute the user interest on that set using the DOI function, receiving our first interest values (① in Figure 2). So far, these represent the entire visualized dataset, meaning that we can already tailor the visualization based on it. We train a regression tree model of the current user interest on that data, using the interest values as labels (②). This model provides us with a first approximation of the user interest on the full dataset, and we can now use it to visualize other interesting data.

Prioritizing stored data When selecting the next chunk of stored data, we want to ensure that this new data is relevant, i.e., our DOI function assigns a high value to it. While we cannot compute the DOI function over the entire stored dataset, we can use the partitioning of the regression tree to focus our selection on those items that it predicts to be relevant. To do so, we translate the decision rules along the path from the root node to leaf nodes with the *highest* interest score into a Boolean filter, similar to our prior work steering-by-example [13]. For example, the query for the leaf node leading to a predicted interest of 0.9 in step ③ of Figure 2 is $(a > 0.5\ AND\ b > 0.15)$. Thus, based on the data currently in the visualization, we predict that items that match this filter are more relevant to the analyst than the rest of the data.

Selecting supplemental data Computing the user interest exclusively over the data we just selected would mean that the result is only valid relative to that new data, but not relative to data that analysts are already familiar with. In other words, items that are most relevant in the context of one set of items are not necessarily as relevant in another set, and could lead to irrelevant data being visualized. To overcome this, we use the interest model to select visualized data to supplement the DOI computation. Since the visualization at this point contains at least c_{max} items, we cannot include *all* visualized data in the computation of the next chunk, so we need to find a smaller representation of it. To make that selection, we again use the regression tree model to retrieve data based on its predicted interest. In contrast to the previous step, though, we now select items from all leaf nodes except for those high-interest nodes, and we also select items from the visualized data this time (④). This essentially results in a balanced sample across leaves of the regression tree, ensuring that our next interest computation is based on data from all degrees of interest seen so far.

Retraining the interest model Now that we have a selection of c_{max} input items, we simply use the DOI function to compute their user interest (⑤). This has two effects: we have computed the user interest for new, previously stored items, and we re-computed the user interest for the visualized data that have been chosen as supplement (⑥). New interest values can be used to tailor the visualization of the new data. The re-computed interest allows us to detect changes to our previous interest model by comparing the old and new interest values for the visualized data in the computation. If the two do not significantly differ, this means that the visualized interest values are still valid. If they do differ, we need to update them. To reduce the cost of this update, we again use the regression tree. That is, we simply measure the change in computed interest per leaf node of the old model rather than overall, and then only update the interest for data selected by each leaf node exhibiting significant changes (⑦). To avoid the overhead of again computing appropriate supplement for computing the DOI function, we instead predict the interest of outdated leaf nodes using the new interest model. We can either perform this update synchronously with the next interest computation (by reducing c_{max}), if the number of items to be updated is relatively small, or asynchronously for more "costly" cases so as not to not impact the human latency constraints of the DOI function.

Updating outdated interest values Aside from changes in the visualized data, there are also other reasons why our user interest model for progressive visualization may become outdated. We can broadly group these reasons based on whether analysts directly adjusted the DOI function or not. In the former case, analysts may invalidate the interest model by, for example, adjusting the terms used in the DOI functions, the weights between them, or their parameters. In the latter case, analysts interactions with the data invalidate the model, for example, by selecting a new focus dataset of interest (see the *DIST* term used by Gladisch et al. [7]), or by simply covering different parts of the data during exploration (see the *KNOW* term used by Gladisch et al. [9]).

The workflow outlined above implicitly captures these changes in the DOI function and will update the computed DOI values in the next step of the progression. This is because a new model will be trained on the adjusted DOI function, which computes a different interest than the model trained on the old DOI function. As the update step of the workflow compares the predictions of the new interest model with the previous model, all items for which the interest changed significantly are "flagged" for update. As a result, our workflow accounts for dynamic adjustments to the DOI function.

4 Evaluation

Setup Following Munzner's nested model [17], we benchmark our approach on the 2018 NYC taxi dataset containing about 112M rows with 17 numeric and categorical columns describing cab rides in New York City from 2018[1]. We limited our benchmarks to 1M rows per test case to be able to fit all data into memory for computing the ground truth and to keep runtimes between tests manageable. A simple task on this dataset is to find outlier taxi rides, which, for example, earned the driver more than usual or because they took unusually long. The following DOI function captures this interest for an item x: $f(x) = norm(x_{total\_amount} + x_{tip\_amount} + x_{trip\_duration} + = x_{trip\_distance})$, with *norm* normalizing the result to $[0,1]$. We compute this DOI function using our approach and using a baseline test case that naïvely computes the DOI function progressively, i.e., it computes the interest for every item only once, at the time when it is added to the visualization. As parameters, we used a chunk

[1] https://data.cityofnewyork.us/Transportation/2018-Yellow-Taxi-Trip-Data/t29m-gskq

size of 1,000 new items per iteration (prioritizing the top 2 most interesting leaf nodes) and a supplement size of 2,000 items per iteration for our approach. We implemented everything in Python 3.8 with pandas and numpy, using the sklearn implementation of regression trees with a maximum depth of 3, computed on standard laptop hardware (i7-8550U with up to 4 GHz, 16 GB RAM). We make the implementation of our approach as well as notebooks for reproducing our benchmarks available on Github[2].

Metrics We evaluated the two test cases to assess (a) how they perform in selecting relevant data for the visualization and (b) how well they avoid large deviations in the predicted interest. To assess (a), we identified the top 1,000 most interesting items in the ground truth, and for each test case count how many of these items are in the progressive visualization at each iteration. For (b), we measured descriptive statistics (min, mean, max, std) of the error compared to the ground truth interest at the end of the progression. We also measured the per-iteration runtime for each test case.

Results Results are summarized in Figure 3. For (a), our approach clearly outperformed the baseline, already retrieving all top-k items within 433 chunks, indicating the benefits of the prioritization step. For (b), our approach performed similar to the baseline in terms of the mean error (baseline: 0.038, ours: 0.017), while reducing the standard deviation (baseline: 0.038, ours: 0.01) and maximum error (baseline: 0.44, ours: 0.39). It also clearly decreased the occurrence of errors larger than 0.1 (baseline: 59,845, ours: 624), i.e., errors more noticeable in a visualization. This indicates that our model was able to detect and correct larger DOI errors through the update step, while the baseline — lacking an update mechanism — did not. Another interesting finding for (b) is that both test cases almost exclusively overestimated the interest, meaning that interest computed progressively is greater than interest computed non-progressively (apparent in the histograms). We explain this as an effect of the DOI function we used, which is affected by rare (i.e., unlikely to be sampled), high-valued outliers that reduce the interest of all other data. In terms of the runtime our approach increased median computation times from 2.04s for the baseline to around 2.06s (mean training times 0.0038s for 1k items, mean query times 0.0019s for 2k items). Both test cases significantly outperformed the non-progressive computation at 97.43s regarding the time until the first results were available.

While these results overall are promising regarding the use of DOI function in progressive visualization, they only cover a static scenario using a preset DOI function with fixed parameters. More work is clearly necessary regarding their impact on interactive analysis scenarios, in which a user actively works with the dataset and dynamically adjusts the DOI function.

5 Implications and Applications

Modeling user interest with regression trees There are several additional benefits in modeling the user interest on progressive visualization with regression trees. One is that regression trees have been shown to perform well on small training datasets [18], particularly useful in progressive visualization where chunk sizes are small compared to the overall dataset size. Furthermore, their simple structure means that training and predicting the user interest is generally fast, and that the model is interpretable by analysts, even lending itself to interactive adjustments as shown, for example, in the work by v.d.Elzen et al. [24]. Lastly, user interest is often modeled hierarchically, for example, in feature definition language by Doleisch et al. [3], meaning that regression trees are also a good fit.

There are also drawbacks to regression trees. For example, they generally only support numeric and categorical data, but not textual or graph datasets. They may also not always yield useful splits of the data dimensions, making them inefficient at prioritizing interesting data. They can also be over-sensitive to changes in the training dataset, irrelevant attributes, and noise [20]. While the literature proposes some solutions to addressing these shortcomings [15], regression trees are not "perfect" models and should not be understood as such. Despite of these theoretical limitations, their inherent ability to generate Boolean queries for each leaf node makes regression trees the most practical technique for our approach.

[2] https://github.com/vis-au/prointerest/tree/doi-tree

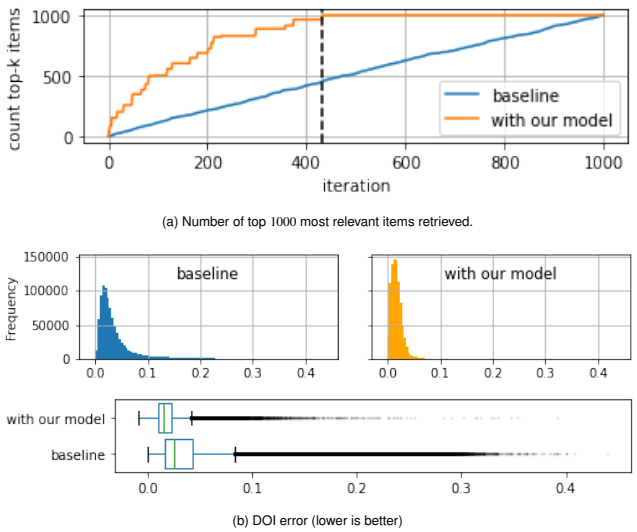

Figure 3: Our benchmarks show our model outperforming the baseline in terms of (a) the number of the most interesting items in the visualization per iteration and (b) the DOI error.

Interest-based tailoring of progressive visualization With the user interest now computed, another consideration is how to tailor the visualization. DOI-enabled concepts widely used in non-progressive visualization like focus+context and information hiding can now also be applied in progressive visualization. Beyond enabling established approaches, we envision dedicated DOI-enabled techniques for the specific challenges of progressive visualization, such as the unique cognitive biases that only arise when visualizing data progressively (see the recent study by Procopio et al. [19]). For example, to alleviate illusion bias ("read something into incomplete results that is not there" [19]) DOI functions can help draw the attention to changes in interesting parts of the visualization whenever new data arrives, automating progressive guards [14]. DOI function can also dampen the visual "buzz" of progressive visualization (like so-called dancing bars [16]), by limiting visual updates to parts of the visualization that are relevant to the analysis task, dampening their effect elsewhere. Thus, we see clear potential for future work in tailoring progressive visualization to the user interest.

6 Conclusion

DOI functions underpin many techniques for tailoring visualizations to analysts' interests, focusing their work on relevant data. As the data relevance changes throughout the analysis, we would need to re-compute the user interest to stay "up to date", which takes longer and longer the bigger the dataset gets. By instead shifting from a computed towards a predicted user interest, our approach allows us to maintain useful interest values even on large datasets by computing the DOI function only on a subset of the data. We see value for this approach beyond the (still relatively young) research field of progressive visualization, as runtime is also a constraint when the entire dataset is visualized at once. Using progressive updates for non-progressive visualization means that analysts can maintain their flow here as well, while a full interest computation would break it. Moreover, having the user interest available progressively opens the door to new techniques for tailoring progressive visualizations.

Acknowledgments

We thank Helwig Hauser and Marc Streit for the fruitful discussions on the topic, as well as the anonymous reviewers for their insightful comments. This work has been funded in part by the Innovation Fund Denmark through the Grand Solution project *Hospital@Night*.

References

[1] J. Abello, S. Hadlak, H. Schumann, and H.-J. Schulz. A Modular Degree-of-Interest Specification for the Visual Analysis of Large Dynamic Networks. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):337–350, 2014. doi: 10.1109/TVCG.2013.109 1, 2

[2] T. d'Entremont and M.-A. Storey. Using a degree of interest model to facilitate ontology navigation. In *2009 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pp. 127–131, 2009. doi: 10.1109/VLHCC.2009.5295284 2

[3] H. Doleisch, M. Gasser, and H. Hauser. Interactive Feature Specification for Focus+Context Visualization of Complex Simulation Data. In *Proc. of VGTC Symposium on Visualization*, pp. 239–248. Eurographics / IEEE, 2003. doi: 10.2312/VisSym/VisSym03/239-248 2, 4

[4] N. Elmqvist and J.-D. Fekete. Hierarchical Aggregation for Information Visualization: Overview, Techniques, and Design Guidelines. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):439–454, 2010. doi: 10.1109/TVCG.2009.84 1

[5] N. Elmqvist, A. V. Moere, H.-C. Jetter, D. Cernea, H. Reiterer, and T. J. Jankun-Kelly. Fluid interaction for information visualization. *Information Visualization*, 10(4):327–340, 2011. doi: 10.1177/1473871611413180 2

[6] J. Fekete and R. Primet. Progressive Analytics: A Computation Paradigm for Exploratory Data Analysis. *CoRR arXiv:1607.05162*, 2016. 2

[7] G. W. Furnas. The FISHEYE view: a new look at structured files. In S. K. Card, J. D. Mackinlay, and B. Shneiderman, eds., *Readings in Information Visualization: Using Vision to Think*, pp. 312–330. Morgan Kaufmann Publishers, 1981. 1, 2, 3

[8] G. W. Furnas. Generalized Fisheye Views. In *Proc. of CHI*, pp. 16–23. ACM, 1986. doi: 10.1145/22627.22342 2

[9] S. Gladisch, H. Schumann, and C. Tominski. Navigation Recommendations for Exploring Hierarchical Graphs. In *Proc. of ISVC*, pp. 36–47. Springer, 2013. doi: 10.1007/978-3-642-41939-3_4 2, 3

[10] H. Hauser. Generalizing Focus+Context Visualization. In G. P. Bonneau, T. Ertl, and G. M. Nielson, eds., *Scientific Visualization: The Visual Extraction of Knowledge from Data*, pp. 305–327. Springer, 2006. doi: 10.1007/3-540-30790-7_18 1

[11] J. Heer and S. K. Card. DOITrees Revisited: Scalable, Space-Constrained Visualization of Hierarchical Data. In *Proc. of AVI*, pp. 421–424. ACM, 2004. doi: 10.1145/989863.989941 2

[12] J. M. Hellerstein, R. Avnur, A. Chou, C. Hidber, C. Olston, V. Raman, T. Roth, and P. J. Haas. Interactive data analysis: the Control project. *Computer*, 32(8):51–59, 1999. doi: 10.1109/2.781635 1

[13] M. Hogräfer, M. Angelini, G. Santucci, and H.-J. Schulz. Steering-by-Example for Progressive Visual Analytics. *ACM Trans. Intell. Syst. Technol.*, 13(6):96:1–96:26, 2022. doi: 10.1145/3531229 2, 3

[14] J. Jo, S. L'Yi, B. Lee, and J. Seo. ProReveal: Progressive Visual Analytics With Safeguards. *IEEE Transactions on Visualization and Computer Graphics*, 27(7):3109–3122, 2021. doi: 10.1109/TVCG.2019.2962404 4

[15] S. B. Kotsiantis. Decision trees: a recent overview. *Artificial Intelligence Review*, 39:261–283, 2013. doi: 10.1007/s10462-011-9272-4 4

[16] D. Moritz, D. Fisher, B. Ding, and C. Wang. Trust, but Verify: Optimistic Visualizations of Approximate Queries for Exploring Big Data. In *Proc. of CHI*, pp. 2904–2915. ACM, 2017. doi: 10.1145/3025453.3025456 4

[17] T. Munzner. A Nested Model for Visualization Design and Validation. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):921–928, 2009. doi: 10.1109/TVCG.2009.111 3

[18] T. Oates and D. Jensen. The Effects of Training Set Size on Decision Tree Complexity. In *Proc. of ICML*, pp. 254–262. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997. doi: 10.5555/645526.657136 2, 4

[19] M. Procopio, A. Mosca, C. Scheidegger, E. Wu, and R. Chang. Impact of Cognitive Biases on Progressive Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 28(9):3093–3112, 2022. doi: 10.1109/TVCG.2021.3051013 4

[20] L. Rokach and O. Maimon. *Decision Trees*, pp. 165–192. Springer US, 2005. doi: 10.1007/0-387-25465-X_9 4

[21] R. Rosenbaum and H. Schumann. Progressive refinement: more than a means to overcome limited bandwidth. In *Proc. of VDA*, vol. 7243, pp. 145–156. International Society for Optics and Photonics, SPIE, 2009. doi: 10.1117/12.810501 1

[22] H. Stitz, S. Gratzl, W. Aigner, and M. Streit. ThermalPlot: Visualizing Multi-Attribute Time-Series Data Using a Thermal Metaphor. *IEEE Transactions on Visualization and Computer Graphics*, 22(12):2594–2607, 2016. doi: 10.1109/TVCG.2015.2513389 2

[23] C. D. Stolper, A. Perer, and D. Gotz. Progressive Visual Analytics: User-Driven Visual Exploration of In-Progress Analytics. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1653–1662, 2014. doi: 10.1109/TVCG.2014.2346574 1

[24] S. van den Elzen and J. J. van Wijk. BaobabView: Interactive construction and analysis of decision trees. In *Proc. of VAST*, pp. 151–160, 2011. doi: 10.1109/VAST.2011.6102453 4

[25] F. van Ham and A. Perer. "Search, Show Context, Expand on Demand": Supporting Large Graph Exploration with Degree-of-Interest. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):953–960, 2009. doi: 10.1109/TVCG.2009.108 2

[26] M. Williams and T. Munzner. Steerable, Progressive Multidimensional Scaling. In *Proc. of InfoVis*, pp. 57–64. IEEE, 2004. doi: 10.1109/INFVIS.2004.60 1

[27] E. Zgraggen, A. Galakatos, A. Crotty, J. Fekete, and T. Kraska. How Progressive Visualizations Affect Exploratory Analysis. *IEEE Transactions on Visualization and Computer Graphics*, 23(8):1977–1987, 2017. doi: 10.1109/TVCG.2016.2607714 1