Optimizing Hierarchical Visualizations with the Minimum Description Length Principle

Rafael Veras and Christopher Collins

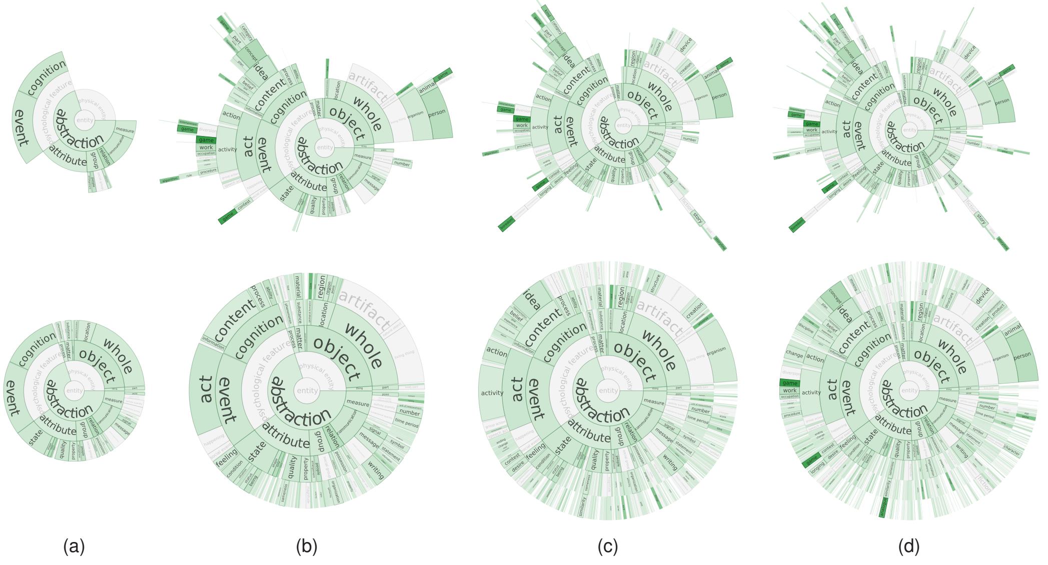


Fig. 1. Display-optimized MDL tree cuts (top row) reveal important nodes while reducing clutter when compared to simple depth thresholding (bottom row). These are Docuburst views of the book *Gamer Theory*, revealing occurrences of concepts in the book using color. Concepts are organized into a semantic hierarchy. From left-to-right increasing numbers of nodes are revealed through adjusting the tree cut parameter (top) or increasing the depth threshold (bottom). We can see that in (b, top) several important dark-green nodes are revealed while unimportant nodes remain hidden. Figure (b, bottom) has twice as many nodes, but important nodes remain hidden. These nodes are not revealed by the simple depth threshold until (d, bottom), where there are a significant number of unimportant nodes also visible.

Abstract—In this paper we examine how the Minimum Description Length (MDL) principle can be used to efficiently select aggregated views of hierarchical datasets that feature a good balance between clutter and information. We present MDL formulae for generating uneven tree cuts tailored to treemap and sunburst diagrams, taking into account the available display space and information content of the data. We present the results of a proof-of-concept implementation. In addition, we demonstrate how such tree cuts can be used to enhance drill-down interaction in hierarchical visualizations by implementing our approach in an existing visualization tool. Validation is done with the feature congestion measure of clutter in views of a subset of the current DMOZ web directory, which contains nearly half million categories. The results show that MDL views achieve near constant clutter level across display resolutions. We also present the results of a crowdsourced user study where participants were asked to find targets in views of DMOZ generated by our approach and a set of baseline aggregation methods. The results suggest that, in some conditions, participants are able to locate targets (in particular, outliers) faster using the proposed approach.

Index Terms—Hierarchy data, data aggregation, multiscale visualization, tree cut, antichain

For many years, the information visualization community followed Shneiderman's celebrated visual information-seeking "mantra" for design: "overview first, zoom and filter, details on demand" [27]. However, as datasets have grown (and small displays have become more prevalent), "overview first" is increasingly challenging to achieve in an effective way. Overviews of very large datasets are often too high-level or cluttered to reveal anything interesting. The task of iterative exploration and sifting through the data is left to the analyst in the traditional model. This paper introduces a method for optimizing large hierarchical visualizations to fit in constrained screen spaces, effectively creating starting point overviews which are designed to balance the goal of maximum information content with the challenge of reducing clutter and enhancing readability. The work is inspired by Keim's visual analytics process, which starts: "analyze first, show the important" [13]. The critical "analyze first" step is addressed in our work to shape the initial view of the data to reveal important data entities while minimizing clutter, harnessing computing power to create data-driven starting points for analysis. The display-optimized tree cut model we present is parameterized to allow for interactive drill down, as well as presentation of optimized overviews of data.

In addition to the challenge of providing optimized overviews for very large datasets, in many situations, visualizations need to be adaptable to a variety of screen sizes. For example, consider an interactive

• *Rafael Veras and Christopher Collins are with University of Ontario Institute of Technology. E-mail: rafael.verasguimaraes@uoit.ca; christopher.collins@uoit.ca.*

Manuscript received 31 Mar. 2016; accepted 1 Aug. 2016. Date of publication 15 Aug. 2016; date of current version 23 Oct. 2016. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TVCG.2016.2598591

visualization embedded as part of an online news story — one version may be appropriate for a smart phone display, while another will be appropriate for a large monitor. The situation is not as simple as changing the zoom factor, or the flow of the webpage, but rather the *level of abstraction* must adjust to make the visualization readable and aesthetically pleasing across devices.

Many factors influence the ability of visualization systems to effectively display large amounts of data; in particular, the available display size, which is determined by the physical constraints of the screen, and the perceptual scalability of the visualization, which depends on the choice of visual representation and layout [33]. Most information visualizations become over-cluttered when the dataset is large. Clutter reduction is an active area of research in information visualization, as elaborated by Ellis and Dix in their taxonomy of clutter reduction methods [10]. Clutter is shown to have a negative impact on visual search [12, 25, 30] and short term memory [20]. In a study of orientation judgment, Baldassi et al. found that clutter causes an increase in orientation judgment errors, and increase in perceived signal strength and decision confidence on erroneous trials [5]. Rosenholtz et al. [26] include the notion of performance in the very definition of clutter: a state in which excess items, or their representation or organization, lead to degradation of performance at some task. Besides, in some resource-constrained client environments (e.g., web browser), the number of graphic primitives necessary to represent large data affects rendering and, consequently, interactive tasks, such as selection and filtering.

In visualizations of hierarchical data, one can take advantage of the hierarchical structure to abstract data at varying levels, in order to reduce the level of clutter when the available space prevents depiction of the full data. Visualizations that implement such strategy are called multiscale visualizations [11] and deciding the appropriate level of abstraction for them is not trivial. Overly-detailed views have high clutter, whereas overly-abstract views can hide important patterns. The right level of abstraction depends on the dataset and the available display space; for example, large desktop displays afford more detail, while mobile phones have not only less space, but also coarser interaction resolution due to the fat finger problem. In this paper, we refer to this problem as the *level of abstraction* problem.

Our display-optimized MDL tree cut technique can be applied to any hierarchical dataset where there are quantitative data values associated with the leaves of the tree. In this paper we will introduce the mathematical foundation behind our general display-optimized tree cut, and demonstrate the approach applied to two popular hierarchical visualization types — treemap and sunburst. Furthermore, we report on multiple validation approaches: a crowdsourced study in which we found that the tree cut approach provides for faster target finding compared to traditional approaches, and a quantitative comparison of clutter and information content across traditional techniques and our display-optimized MDL treemaps.

1 Related Work

In this section, we survey two areas: techniques for controlling clutter in visualizations using aggregation and the use of tree cuts (also known as antichains) to navigate large graph hierarchies.

1.1 Clutter Control

Based on the cartographic principle of constant information density [28], VIDA is a system that automatically creates visualizations in which density remains constant across zoom levels (z dimension) and within each view (x and y dimensions) [31]. The display is divided into regions, where the visual representation is modified (e.g., dots instead of glyphs) to meet a target density value specified by the user. Density measures are number of objects and number of vertices per unit of display area.

ViSizer is a framework for resizing visualizations [32]. It employs a sophisticated image warping technique that scales important regions uniformly and deforms less important regions. The significance measure is a compound of Rosenholtz et al.'s perception-based clutter measure [25] and a degree of interest (DOI) function. ViSizer focuses on non-space filling visualizations such as word clouds and scatterplots.

Chuah [7] employs a simple strategy for automatic aggregation in histograms, and ordered radial and treemap visualizations: aggregate neighboring objects whenever there is occlusion or they are too small to be perceived. This approach works better where data items have an intuitive order (e.g., time series, histograms, or file directories ordered by name). Cui et al. [9] tackled the optimal level of abstraction problem, but focusing only on accuracy; that is, how well the abstracted data represent the original dataset. They proposed two measures of quality: the histogram difference measure and the nearest neighbor measure, which were integrated into XmdvTool. As the measures do not account for the visual quality of the resulting visualization, the user determines the best view interactively, by tweaking the level of detail and comparing the quality measure values. Likewise, based on aggregation quality measures, Andrienko and Andrienko [2] allow users to specify the desired level of abstraction in visualizations of movement data (flow maps).

Koutra et al. [14] proposed a parameter-free method based on the minimum description length to select the best (most succinct) summary for large graphs among a set of alternatives: cliques, stars, chains, and bipartite cores.

Perhaps the closest to this proposal, Lamarche-Perrin et al. [15, 16] introduce a method for selecting abstract representations of hierarchical datasets. In their work, a two-part information criteria consisting of entropy and Kullback-Leibler divergence is used to select the tree cut featuring the best balance between conciseness and accuracy. Their procedure requires tuning a free weighting parameter that specifies the relative importance of one criterion over the other. It does not account for the available display space, so any adjustments to accommodate small or big screens need to be done manually by tuning the aforementioned weighting parameter.

1.2 Tree Cuts or Antichains

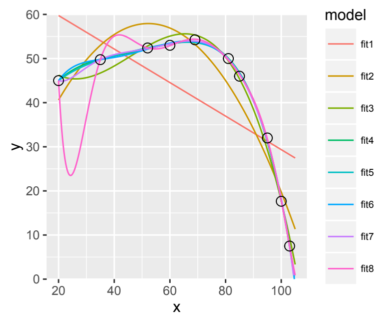
Tree cuts, also known as antichains, have been widely used in the exploration of large graphs and hierarchies. SentireCrowds [6] and ThemeCrowds [3] employ a maximal antichain selection method to abstract a hierarchy of topics visualized as a treemap. That method is based on matching node scores resulting from user queries. GrouseFlocks [4] reduces the complexity of interacting with large graphs by letting users manipulate cuts of superimposed aggregate hierarchies. Users can adjust the cut level of abstraction by performing topology-preserving operations involving merging and deletion of aggregates. In order to ensure the abstracted hierarchy view remains under the display capacity, ASK-GraphView [1] parametrizes clustering with maximum antichain size. In ASK-GraphView and GrouseFlocks the hierarchies are not part of the data, but created by an algorithm. This allows great flexibility to modify the hierarchy structure around display constraints. In this work, we focus on "rigid" hierarchies, where classes carry domain specific relevance and, thus, cannot be merged or deleted without cost to interpretation.

2 Theoretical Foundations

Suppose a set of measurements $D = (x_1, y_1), .., (x_n, y_n)$ was collected as part of an experiment and we were asked to send this data over a network where the transmission cost is high. Among the countless possible ways of encoding the data, it is in our best interest choosing a scheme that allows for the shortest message. In this scenario, the code length for sending the *raw* data, assuming that encoding a number has a fixed cost of b bits is:

$$L(D) = \sum_{i=1}^{n} \Big\{ L(x_i) + L(y_i) \Big\} = 2nb. \quad (1)$$

If the relation between x and y can be described by a polynomial model (or any other model), it might be possible to reduce significantly the code length. As an example, let's examine the polynomial case. A



	$\sum L(\hat{a}_k)$	$\sum L(r_i\|\hat{\theta})$	total
1	128	640	768
2	192	640	832
3	256	512	768
4	**320**	**128**	**448**
5	384	192	576
6	448	192	640
7	512	128	640
8	576	0	576

Fig. 2. On the left, a series of polynomials ranging from order 1 to 8 fitted to a 10-point data set. On the right, the cost of encoding (in bits) the two parts of each polynomial model.

polynomial regression model has the following form:

$$\hat{y} = \sum_{k=0}^{p} \hat{a}_k x^k + \varepsilon. \tag{2}$$

So the code length of the data as seen through a fitted polynomial model $\hat{\theta}$ is:

$$L(\hat{\theta}, D) = \sum_{i=1}^{n} L(x_i) + \sum_{k=0}^{p} L(\hat{a}_k) + \sum_{i=1}^{n} L(r_i \mid \hat{\theta}), \tag{3}$$

where $\hat{a}_k$ is the k-th parameter of the polynomial and r_i is the i-th residual. Namely, the equation above is a sum of the cost of encoding the model and the cost of encoding the data conditioned on the model (residuals). Note that the cost of sending the vector x is constant across all models. As a polynomial might not fit the data perfectly, it is necessary to send the model residuals, so that the receiver is able to reconstruct D accurately. However, depending on our tolerance to errors, we might be willing to ignore residuals smaller than a fixed threshold. The better the fit, the more economical is the description of the residuals. Overall, it is only worth representing our data with a polynomial model if we can find a model whose code length overhead is smaller than the code length of vector y:

$$\sum L(y_i) > \sum L(\hat{a}_k) + \sum L(r_i|\hat{\theta}). \tag{4}$$

To illustrate this notion, consider the ten data points depicted in Figure 2, left. We fitted to this data a family of polynomials of increasing order and compared the cost of representing the data with each of them in a setting where any number is represented with 64 bits, and residuals smaller than 0.5 are ignored. Figure 2, right, shows the cost of each fitted polynomial from order 1 to 8. It is clear that the more parameters a model has, the better is its fit. However, the model that provides the shortest description is that featuring the best balance between goodness of fit and complexity. In our example, this model is the 4th order polynomial, which also satisfies (4), as the cost of encoding y in the naive scheme is 640 bits.

In this example, we used information theoretic tools to determine the model that most concisely captures the regularities in the data. The criterion we employed is a simplification of the Minimum Description Length (MDL) Principle, which we describe formally in the following subsection. MDL is a powerful approach to model selection that has been used to solve a large variety of problems, including polynomial regression, Gaussian density mixtures and Fourier series regression [17], and applied problems such as image segmentation [18], learning word association norms [19] and learning decision trees [22].

2.1 Minimum Description Length

Proposed by Rissanen, MDL is an information criterion used for model selection in statistics [23]. The principle is based on the following notion: given a set of observed data and a family of fitted models, the best model should provide the shortest encoding of the data. The description length of a model is calculated as a sum of two parts: the length of the binary codes that describe a) the model parameters, and b) the data residuals [18]. More formally, the MDL criterion can be written as:

$$L(\hat{\theta}, x) = L(\hat{\theta}) + L(x \mid \hat{\theta}), \tag{5}$$

where $\hat{\theta}$ is a parameter vector, x is the data, and $L(\hat{\theta})$ and $L(x \mid \hat{\theta})$ are the parameter description length (a) and the data description length (b), respectively.

Unlike in the polynomial example, where we used computer-oriented calculations for the code length, MDL is concerned with *optimal* code length. That is, with MDL, we do not care about how a model is encoded in practice as much as we care about how concisely it can be encoded in theory. Let A be an alphabet and α be any of the symbols in A. If the probability $p(\alpha)$ of occurrence of $\alpha \in A$ is known, then in the optimal encoding scheme for A the length of α is:

$$L_{OPT}(\alpha) = -\log_2 p(\alpha). \tag{6}$$

This result is important because often the likelihood function of the model $\hat{\theta}$ is known, so the data description length (number of bits to encode the residuals) follows from (6):

$$L(x \mid \hat{\theta}) = -\log_2 p(x \mid \hat{\theta}). \tag{7}$$

For instance, in our polynomial example we could leverage the fact that, as per the regression model assumption, the residuals are approximately normally distributed, and use the log of the Gaussian likelihood, given by $(n/2)log_2(RSS/n)$, as $L(x \mid \hat{\theta})$, where RSS is the residual sum of squares.

Frequently, the probability distribution of the model parameters (usually, a vector of integer or real numbers) is not given; in this case, Rissanen [23] proposes a *universal prior* probability distribution or, equivalently, a coding system, for integers. Rissanen demonstrated that the optimal code length for such integers with unknown probability function can be achieved with his coding system and approximated to $log_2 n$. Therefore, we can estimate the description length of arbitrarily complex models, as long as their parameters can be described as arrays of integers or real numbers.

Let's assume $\hat{\theta}$ is a vector of real numbers, which can be encoded by representing the integer and fractional parts separately. The fractional part needs to be truncated to a pre-defined binary precision ρ, since the binary representation of many numbers can be infinite. Thus, the number of bits to encode $\hat{\theta}$ is:

$$L(\hat{\theta}) = \sum_{i=1}^{k} \log_2 \lfloor \hat{\theta}_i \rfloor + k\rho, \tag{8}$$

where k is the number of parameters in the model.

Note that the choice of the precision ρ is of major importance. Choosing fewer bits to encode the fractional parts yields a small $L(\hat{\theta})$, but at the expense of $L(x \mid \hat{\theta})$, as the residuals will be larger. A finer precision reduces the residuals, as the encoded values will be closer to the true estimates, but increases the cost of encoding the parameters. In order to minimize the description length, we need to optimize the precision. Rissanen [24] shows that if the model parameters are estimated from n data points using Maximum Likelihood Estimation (MLE) and n is large, the optimal precision ρ is approximately $(\log_2 n)/2$. Thus, (8) can be rewritten as:

$$L(\hat{\theta}) = \sum_{i=1}^{k} \log_2 \lfloor \hat{\theta}_i \rfloor + \frac{k}{2} \log_2 n. \tag{9}$$

With the expressions for data and parameter description length, (5) can be written in more detail as:

$$L(\hat{\theta}, x) = \sum_{i=1}^{k} \log_2 \lfloor \hat{\theta}_i \rfloor + \frac{k}{2} \log_2 n - \log_2 p(x \mid \hat{\theta}). \tag{10}$$

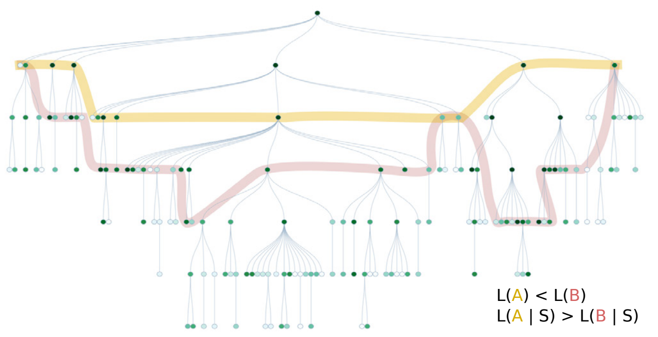


Fig. 3. An illustration of two tree cuts. More abstract cuts have lower parameter description length, but higher data description length.

Equation 10 embodies the fundamental trade-off between conciseness and accuracy that defines the MDL principle. Models with more parameters will achieve better accuracy (high likelihood) at the expense of simplicity. In fact, if we set $L(\hat{\theta})$ constant, MDL falls back to MLE, selecting the model that offers the best fit to the data. In that sense, $L(\hat{\theta})$ can be thought of as a safeguard against over-fitting. Likewise, over-concise models have low accuracy, being just as undesirable. Minimization of the description length tends to select the model featuring the best balance between these criteria. In the information theoretic interpretation, the selected model corresponds to the best compression of the data.

2.2 MDL Tree Cut Model

Having laid out the general formulation of the MDL principle, in this section we explain the tree cut model, which is an important building block for our abstraction approach. The tree cut model is a generalization method based on MDL, originally developed for the linguistic problem of automatic acquisition of case frame patterns from large corpora [19].

Consider a tree structure representing the hierarchical relation between abstract classes, e.g., IS-A, part of, instance of. The degree of abstraction grows towards the root. Assume that only the leaves are observable (countable), and the internal nodes accumulate the counts of their children. L is the set of all leaves. A dataset S is a multiset of observations, each representing one occurrence of a leaf $l \in L$, with $l \in S$ denoting the inclusion of l in S as a multiset. We denote the dataset size by $|S|$, the total number of observations.

A tree cut is any set of tree nodes that exhaustively covers the leaf nodes. Graphically, it can be represented by a path crossing the tree lengthwise, as in Figure 3. Nodes along the cut represent the subtrees dominated by them and are assigned each a probability value. Depending on how regular is our data S, a concise way to transmit it over an arbitrary channel to a receiver who has knowledge of the tree is to send a tree cut. The receiver then estimates the value of each leaf based on the value of the node representing it in the cut. In other words, a cut is a model of the data, carrying estimates of the observed values. The residuals are sent separately, in the MDL fashion, as discussed in Section 2.1.

A tree cut model M is defined as the tuple $(\Gamma, \hat{\theta})$, where $\Gamma = [C_1, C_2, ..., C_k]$ and $\hat{\theta} = [\hat{P}(C_1), \hat{P}(C_2), ..., \hat{P}(C_k)]$: the vector of nodes (classes) and their parameters (estimated probabilities), respectively. The probability $\hat{P}(C)$ of a class is estimated by MLE, as follows:

$$\hat{P}(C) = \frac{f(C)}{|S|}, \tag{11}$$

where $f(C)$ is the accumulated count of the class C. The estimated probability of each of the leaves under a class is obtained by normalization of the class probability over the number of leaves C under the class:

$$\hat{P}(l) = \frac{\hat{P}(C)}{|C|}. \tag{12}$$

Note that behind this formula is the assumption of uniform probability. This means the probabilities (or frequencies) of the leaves under a cut are smoothed, reflecting the decrease in uncertainty when data is represented by abstract models.

As discussed in the previous section, the data description length is the log of the likelihood of the data:

$$L(S \,|\, \Gamma, \hat{\theta}) = -\sum_{l \in S} \log_2 \hat{P}(l). \tag{13}$$

The minimum data description length is held by the deepest tree cut model, comprised of all leaves, which features no better abstraction than the raw data. The cost of encoding the parameters $\hat{\theta}$ of the model, an array of real numbers, is given by (9). Note that, Li & Abe omit the first term in (9), namely, the cost of encoding the integer part of the parameters, because the model parameters are probabilities; hence, the cost of encoding the integer parts is always 0. In summary, Li and Abe's tree cut model minimizes the following information criterion:

$$L(\hat{\theta}, S) = \frac{k}{2} \log_2 |S| - \sum_{l \in S} \log_2 \hat{P}(l). \tag{14}$$

To be more precise, in addition to the probabilities $\hat{\theta}$, a receiver would also need to know the classes Γ to decode the data correctly. Since the number of possible cuts in a tree is finite, in theory we could use an index to inform Γ, as part of the coding scheme. As such indexes would be equally probable a priori, their code length would be constant for all models and so, can be safely ignored. For sake of model selection, all we need to account for is the cost of encoding $\hat{\theta}$ and $(S \,|\, \hat{\theta})$.

Li and Abe provided an efficient and simple bottom up algorithm, based on dynamic programming, that is guaranteed to find the tree cut whose description length is minimal [19]. In the rest of this paper, we present different ways to calculate parameter and data description lengths, but the same algorithm is used for minimization.

3 MDL Drill-Down

In this section, we experiment with using the tree cuts selected by Li and Abe's approach to inform which nodes should be abstracted in views of a hierarchical dataset. Since such cuts are generated with no consideration of the available display size, we adapt the method by introducing a weighting parameter that determines the relative importance of fitness to the data over clutter. An increase in weight results in a deeper tree cut. In our proof-of-concept, the user can manipulate this parameter seamlessly through conventional drill-down to increase the level of detail of the view.

We chose to implement the technique on Docuburst, an open-source document visualization tool [8]. Docuburst displays a sunburst representation of the WordNet ontology where the size of nodes and categories (angular extent) is weighted by their occurrence in the input document, allowing users to inspect which words and categories of words are more prevalent in a document. The color of a node is based on the non-cumulative count of uses of the corresponding word in the document. In their future work section, Collins et al. discuss two problems that could potentially be solved with uneven MDL tree cuts. First, abstracting subtrees that have low relative importance. Second, the top levels of WordNet are too abstract, as far as carrying little information about the document's content.

Figure 1 features views of the book *Gamer Theory*, by McKenzie Wark. The most representative categories of the document are the darkest (most frequent); for instance: game, entertainment, algorithm, storyline, boredom, etc. In a full tree view, 6,302 nodes would be rendered, which is likely enough to cause latency in a browser-based visualization. Also, displaying this many nodes results in small, illegible labels and the need to interactively zoom and pan.

The tree cut resulting from minimizing Li and Abe's information criterion is shown in Figure 1(a). Nodes under the tree cut are hidden, whereas nodes on or above the tree cut are visible. Unless the available display size is limited, that view can be considered too abstract.

Following Wagner [29], we introduce a free weighting parameter W to equation (14) as a means to control the importance of the data description length over the parameter description length and, as a result, the tree cut depth:

$$L(\hat{\theta},S)=\frac{k}{2}\log_2|S|-W\sum_{l\in S}\log_2\hat{P}(l) \quad (W>0). \tag{15}$$

The semantics of increasing W is equivalent to that of drilling down; the more weight applied to the data description length (residuals), the more parameters (nodes) will be included in the model (tree cut), to minimize the overall description length. Thus, weighted MDL tree cuts can be useful to reveal details at a rate that is more compatible with the distribution of values in the hierarchy. In order to illustrate this concept, we mapped W to the drill-down action in Docuburst; that is, when users roll the mouse wheel, W is incremented/decremented by a predefined delta. Figure 1(b-d), *top*, shows the result of three subsequent increments in W, starting from 1(a), top. In contrast, Figure 1(b-d), *bottom*, shows the result of three drill-down steps where a conventional depth threshold is incremented. It is clear that, in only a few steps, weighted MDL views allow access to most of the representative nodes in the document with much less clutter than using the depth threshold or the full overview. In terms of number of nodes rendered (a-d), the weighted MDL views cost 32, 387, 730, and 887 nodes; while the depth threshold views cost 183, 808, 2202, 4199 nodes.

An important concern is choosing ΔW so that every increment results in a view that has significantly more information than the previous. In our tests, ΔW was defined empirically, and a value of 250 yielded good results for visualizing a variety of documents.

Weighted MDL views can be useful as a way to explore visualizations interactively, but the problem of optimizing the level of detail as a function of the available display space *before any user input* remained unsolved. Specifically, we needed a method capable of generating a *first* view of the dataset that is as informative as possible within the bounds of readability. Next section presents a satisfactory method.

4 Display-tailored tree cut models

We begin this section with the consideration that hierarchical visualization concerns, in general, the representation of tree cut models, in the sense defined in Section 2.2. If we treat visualization techniques (e.g., treemap, sunburst) as coding schemes and the views produced with them as encoded tree cut models, we can select optimal views using MDL criteria. In particular, we are interested in expressing parameter and data description lengths in a way that relates to clutter and fitness in visualizations. We will focus on space-filling hierarchical visualization techniques, as the connection to MDL is more obvious.

In a space-filling visual representation of a hierarchical dataset, the pixel grid is divided into areas proportional to the data values. Areas are recursively grouped in the visual space according to the hierarchy topology, so that siblings are always adjacent. In addition, color and labels can be used to convey the hierarchical structure.

A non-aggregated hierarchical visualization V_{max} is an encoding of the *deepest* tree cut model of a dataset. For example, in a treemap without decorations (e.g., padding), V_{max} fills the entire display space with rectangles representing the tree leaves. Given the set Λ of visualizations of S using a specific layout, each of which corresponds to a tree cut of S, V_{max} is the visualization that maximizes $L(V)$:

$$V_{max}=\underset{V\in\Lambda}{\arg\max}(L(V)), \tag{16}$$

where S denotes the dataset. Note that, for sake of simplicity, we make no distinction in the notation between a visualization V and the tree cut encoded by it.

In the information theoretic interpretation, if visualizations allowed for lossless coding, V_{max} would always minimize $L(S\,|\,V)$ and provide the best fit to the data, corresponding to the model selected by MDL when we set $L(V)$ constant or, equivalently, to the model selected by MLE. However, a space-filling visualization is a *partial* and *lossy* coding system: partial because there exist some source symbols that cannot be encoded (e.g., data points that map to subpixel areas); lossy because it is possible that a pair of symbols share a code word (e.g., data points that map to overlapping areas due to rounding).

Depending on the available display space, when the dataset is relatively small, V_{max} generally provides the best fit to the data, but when the number of leaves is large, decoding of information is impacted, due to the aforementioned limitations caused by display pixel resolution. This is a key departure from Li and Abe's method, where an increase in the length of the model always yields an increase in fitness. In other words, there is a limit on the model fitness to data achievable by a space-filling visualization. The fact that V_{max} does not necessarily hold the minimum data description length can be denoted as follows:

$$L(S\,|\,V_{max})\geq\min_{V\in\Lambda}L(S\,|\,V), \tag{17}$$

This inequality can be read as: the data description length of the visualization of the deepest tree cut (V_{max}) is not necessarily minimal. As a result, before even considering the parameter description length, we can observe that it pays off selecting treemaps more abstract than V_{max} when datasets are large relative to the available screen size.

4.1 Treemap

Let's define the dataset S in more detail. S is a 2-tuple (L,f), where L is the subset of classes that are tree leaves, and f is a function such that for each $l\in L$, $f(l)$ is the count of l.

Then the area of a leaf can be defined as the following composite function with respect to the display area D (in pixels):

$$(A\circ f)(l)=A(f(l))=\frac{f(l)}{|S|}D. \tag{18}$$

Likewise, the area of an abstract class C is given by:

$$A(f(C))=\sum_{l\in C}A(f(l)), \tag{19}$$

where $l\in C$ is the set of tree leaves dominated by C. We call $G=(L,A\circ f)$ the linearly transformed dataset using $A\circ f$. Essentially, G is the dataset with scores transformed to pixels. The probabilities of the classes are estimated based on the encoded G. For conciseness, we abbreviate $A(f(C))$ as $A(C)$ in the rest of this section.

A treemap encodes such areas as a vector of rectangle coordinates $R=[R_1,R_2,...,R_k]$. Formally, we describe a treemap as a 2-tuple $\hat{T}=(\Gamma,(R\,|\,D))$. Given D, we can refer to any point in the grid with an integer index $1\leq i\leq D$. Thus, to transmit R_i, we need only two integers, corresponding to the indexes of the top left and bottom right corners. Since the index space is finite and the indexes are equally likely, we can use Rissanen's universal prior to arrive at $L(i)=\log_2(D)$. The parameter description length is then:

$$L(R)=2k\log_2 D. \tag{20}$$

Equation 20 gives an approximation of the *optimal* number of bits necessary to transmit the parameters of a treemap, ignoring any factors that are constant across all treemaps. For the sake of simplicity, we consider a treemap with no colors or labels.

Since the pixel grid imposes a limited precision on the representation of areas, we approximate the encoded area of C in the treemap by rounding $A(C)$:

$$A'(C)=\lfloor A(C)+1/2\rfloor. \tag{21}$$

Note that $A'(C)$ does not account for precision lost by the fact that $A(C)$ has to be decomposable into exactly two factors. $\hat{P}(C)$ is estimated simply as the ratio between the encoded class area and the total display area:

$$\hat{P}(C)=\frac{A'(C)}{D}. \tag{22}$$

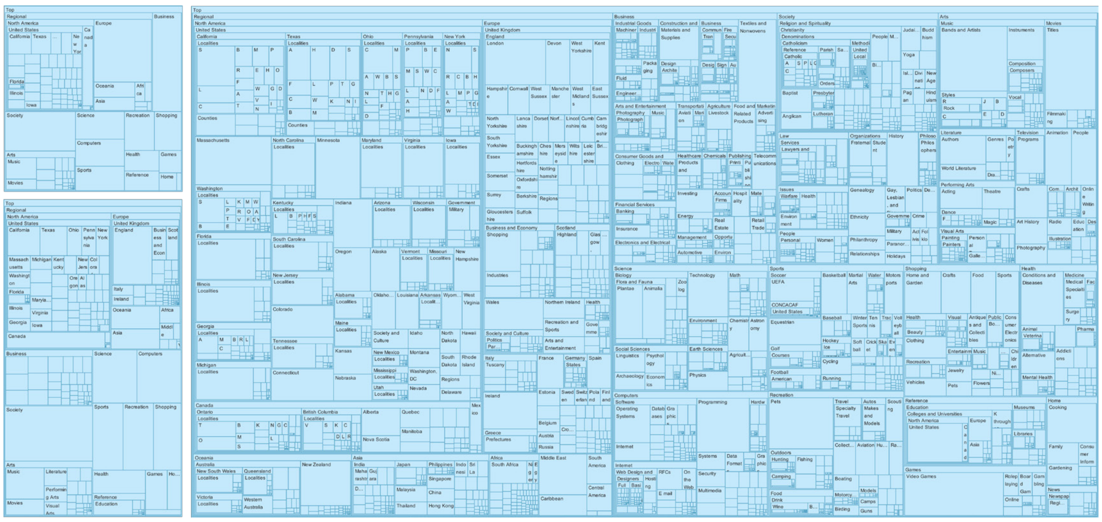


Fig. 4. Treemap visualizations generated with the display-tailored MDL procedure, with the following resolutions: 375x400px, 375x667px and 1920x1080px. More abstract tree cuts are selected for smaller displays.

As in (12), we assume that $\hat{P}(l)$ is estimated by normalizing $\hat{P}(C)$ with respect to the leaves dominated by C:

$$\hat{P}(l) = \begin{cases} \frac{\hat{P}(C)}{|C|} & \text{if } \hat{P}(C) > 0 \\ c & \text{if } \hat{P}(C) = 0 \end{cases} \tag{23}$$

where c is a constant representing the estimated probability of the leaves under a class whose rounded area is zero, and be thought of as an uninformed probability. We can set c to an arbitrarily small value so as to penalize cuts featuring subpixel areas or, more sensibly, define c as the sum of the probabilities of the "invisible" classes in a cut, divided by the total number of tree leaves under such classes. The piecewise function above can also be defined in a more conservative way; for example, setting $\hat{P}(l) = c$ if $A'(C) < \delta$, in order to penalize cuts with small areas, where δ is the smallest visible area. For example, a 3x3 pixel area on a high resolution wall-sized display.

The data description length is $L(G\,|\,\hat{T})$, the log of the following likelihood of G (as discussed in Section 2.1):

$$\mathscr{L}(G\,|\,\hat{T}) = \prod_{l \in L} \hat{P}(l)^{A(l)} \tag{24}$$

It is worth mentioning that the expression above is not strictly a likelihood, but a power of the likelihood, since the data counts have been multiplied by a common factor that converts them to areas. Finally, the information criterion for selection of treemaps is:

$$L(\hat{T}, G) = L(\hat{T}) + L(G\,|\,\hat{T}) = 2k \log_2 D - \sum_{C \in \Gamma} A(C) \log_2 \hat{P}(l) \tag{25}$$

4.2 Sunburst

The structure of a sunburst can be thought of as a series of overlapping disks, one for each tree level. A tree cut can be represented as a vector of arcs Q. The central angle of the arc of a class equals the sum of its children's angles. Arc radius is proportional to the depth of a class in the tree: $R_j = (j+1)\Delta R$, with R_j being the radius of all classes of depth j, and $\Delta R = D/2(h+1)$, where D is the sunburst diameter and h is the tree height.

It is reasonable to assume that users decode a sunburst by estimating the ratio of the arc length of a class and the circumference of the disk that corresponds to the tree level where the class belongs. Assuming the sunburst is sized to optimally fit the screen, as more levels are displayed, the radius of each tree level's disk is reduced, and estimating the value of a class becomes more difficult. This implies that selecting the best tree cut depends on how many levels are displayed, and vice-versa. For example, a class with a relatively low frequency and depth 2, might be readable when displaying only three levels of a tree, but can be rendered invisible when eight more levels are displayed, as the level disks will shrink.

Consequently, we need to run two rounds of minimization of the description length. In the first, we select the best tree cut under each value of ΔR, restricting the levels of h; in the second, we select the best of the tree cuts from the previous step. This is only valid if we define the encoding of the data in a way that tree cut models are comparable across values of ΔR; that is, they need to encode the same quantities.

We define the following mapping of S, where function A is the area of the arc sector of radius $D/2$, which is independent of ΔR.

$$(A \circ f)(l) = A(f(l)) = \frac{f(l)}{|S|}\pi(D/2)^2. \tag{26}$$

A sunburst needs only two integers to inform each area, corresponding to the pixel indexes of the endpoints of an arc:

$$L(Q) = 2k \log_2 D. \tag{27}$$

The arc length s of a class C at depth j is:

$$s(C) = \frac{f(C)}{|S|} 2\pi R_j. \tag{28}$$

$A'(C)$ is calculated analogously to (21), based on the rounded arc length:

$$A'(C) = \lfloor s(C) + 1/2 \rfloor \frac{(D/2)^2}{2R(C)}, \tag{29}$$

which means that the decoded area of arcs with length smaller than .5 is 0, due to the pixel resolution constraint. The estimated probability of a class is, thus:

$$\hat{P}(C) = \frac{A'(C)}{\pi(D/2)^2} \tag{30}$$

4.3 Proof-of-concept

To illustrate the use of our display-tailored MDL procedure, we developed two prototype visualizations (treemap and sunburst) of the Directory Mozilla (DMOZ) dataset. As of November, 24, 2014, DMOZ consisted of 3,847,266 web pages, categorized under a total of 782,031 topics. We selected the subtree under the prefix "Top/World", which contains 2,083,282 pages written in English under 498,487 topics. We wrote browser-based clients that request tree cuts from a Node.js server. The parameters required by the server are display size and root node ID. The layouts are calculated in the server using D3 and rendered in HTML. Although the server has no knowledge of the algorithm used by the client to calculate the treemap, it relies on the fair assumption that the areas are calculated approximately as Section 4.

The resulting visualizations, parametrized for a variety of screen resolutions, are presented in Figure 4. Note that as the display resolution increases, deeper tree cuts are selected. This is a consequence of fewer classes in such cuts being represented with tiny areas; hence, the likelihood of these cuts increases, while their description length decreases.

The treemaps drawn by our client allocate significant space for labels, in a way commonly known as "padding". That space is subtracted from the space available to represent each node's ancestors, and is also meant to help users understand the tree structure better. Our MDL calculations do not account for this "wasted" space (in the estimation sense) and the clutter introduced by the labels; therefore, there is more complexity in the resulting views than what is encoded in $L(\hat{T})$. We consider, however, the results satisfactory.

5 Validation

This paper is based on the premise that a high-quality display of hierarchical data has a good balance between clutter and information; hence, the main question to be answered is whether the proposed approach is scalable, in the sense that it can consistently produce high-quality views under varying display resolutions and dataset sizes. It should be noted that it is not our intention to provide a comprehensive evaluation of abstraction approaches; instead, we are interested in comparing the proposed method with reasonable baselines to put its quality in perspective.

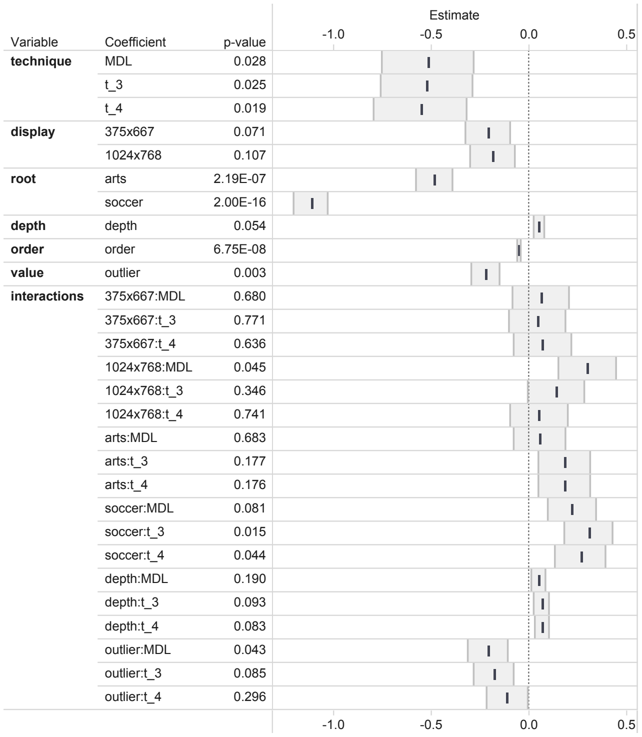
To address this, we adopted two validation approaches, following Munzner's nested model of validation [21]. At the visual encoding level, we test performance in a comparative controlled study, and we report on a quantitative image analysis measuring clutter. Finally, at the algorithm level, we report on the scalability of the approach.

5.1 User Study

Clutter is shown to correlate with response times in visual search tasks [12, 25, 30]; therefore, a sensible way to assess the level of clutter in a visualization is by measuring the time participants take to locate targets. In hierarchical displays, an important caveat of abstraction is hiding potentially interesting nodes; that is, if a node of interest is located deep in the hierarchy, more abstract views will require more drill downs to locate it. We designed a user study where participants were asked to find targets in treemaps abstracted with different methods, including MDL. Among other factors, we varied display resolution, target value, and target depth in the tree, and examined how each abstraction approach performed in interactive tasks.

5.1.1 Tasks

Participants were instructed on how to use the drill down (re-rooting) function and were given the *path* to the target (i.e., a list of the target's ancestors); for instance: *Top/Arts/Music*. We implemented a CSS hack to make labels not searchable with a browser's find tool. The following factors varied in the tasks: abstraction technique, display size, dataset size, target depth, and target value. We compared MDL with three levels of depth threshold: t_3 and t_4, which correspond to the conservative approaches of capping nodes with depth greater than 3 and 4, and t_∞, which is equivalent to no aggregation. Display resolution has three levels: *375x667*, *1024x768*, and *1920x1080*, which match common resolutions of smart phones, laptops, and desktop monitors, respectively. For dataset size, we tested three subtrees of DMOZ: *top*, *arts* and *soccer*, containing approximately 500,000, 55,000, and 3,000 categories each. Target depth (distance from root) varied among 3, 4, 5, and 7; and target value varied between *average* and *outlier*. The value of average targets was the average of the values of all categories in the target's level, while the value of outliers was ten times the average. Given these constraints, the target location in the tree was chosen randomly. Depending on the combination of factors, the target might be visible in the "overview" screen or drill down might be necessary to find it; for example, a target with depth 4 in a treemap where nodes



Variable	Coefficient	p-value
technique	MDL	0.028
	t_3	0.025
	t_4	0.019
display	375x667	0.071
	1024x768	0.107
root	arts	2.19E-07
	soccer	2.00E-16
depth	depth	0.054
order	order	6.75E-08
value	outlier	0.003
interactions	375x667:MDL	0.680
	375x667:t_3	0.771
	375x667:t_4	0.636
	1024x768:MDL	0.045
	1024x768:t_3	0.346
	1024x768:t_4	0.741
	arts:MDL	0.683
	arts:t_3	0.177
	arts:t_4	0.176
	soccer:MDL	0.081
	soccer:t_3	0.015
	soccer:t_4	0.044
	depth:MDL	0.190
	depth:t_3	0.093
	depth:t_4	0.083
	outlier:MDL	0.043
	outlier:t_3	0.085
	outlier:t_4	0.296

Fig. 5. Results from a generalized mixed linear model (Gamma, log-linked) fitted to the user study data. Response variable is completion time. Estimates are in log scale.

with depth higher than 3 are hidden (i.e., t_3) can only be seen upon a drill down. The crossing of all factors resulted in 288 interactive tasks.

During pilot testing, we realized that some tasks might take a long time (over two minutes), and a long session is incompatible with participants' expectations of fairness in crowdsourcing tasks. Thus, each session consisted of one training task followed by 8 tasks. In total, each participant completed 9 tasks, which were assigned randomly based on display resolution, in order to avoid participants having to interact with visualizations larger than their screen. Completion times and number of drill-down interactions were recorded. In order to minimize the effect of latency, in the interactive tasks the timer was paused whenever the user drilled down, and resumed once the new view was completely rendered.

5.1.2 Participants

Participants were recruited with the CrowdFlower crowdsourcing platform and compensated with \$2. They were presented with the instructions both on the CrowdFlower page listing our study and on the study page hosted in our servers. Participants were allowed to skip each task after three minutes and withdraw the study at any time.

5.1.3 Results

We analyzed 980 completed trials ($\sim$ 3.4 per task avg.) after removing 96 outliers. The median session length was 11 minutes. We used a log-linked Gamma generalized linear model, including as covariates display resolution, dataset size, target depth and target value both as main effects and in two-way interactions with technique. A new variable was created representing the order tasks are completed within the session. User was included as a random intercept. Baseline levels are t_∞, 1920x1080px, *top*, average, depth and order 0.

The model intercept is 4.37 (79 seconds). Model estimates correspond to increase/decrease in the intercept estimate, which is in log scale. For instance, for an intercept of 4.37, a variation of -0.1 represents a reduction of 8 seconds in mean time. The results show that, relative to t_∞, all other techniques are responsible for a significant decrease in response times on average (Figure 5). Order has a small, but significant negative effect on times and so does changing the value of the target to outlier, to a larger extent. The outlier effect is significantly

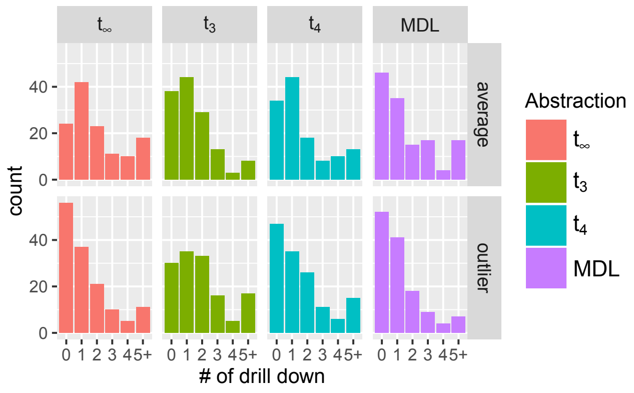


Fig. 6. Number of drill-down interactions needed to complete a single trial of the study, grouped by abstraction technique and target value.

and slightly larger for the MDL approach, although the differences in estimates are not dramatic. Interestingly, depth does not seem to significantly affect the response variable. This may be due to our not accounting for the time for new views to load when drilling down. A decrease in dataset size improves completion times only for t_∞, and in the smallest dataset condition (*soccer*), both t_3 and t_4 perform worse than t_∞. This is probably due to participants having to drill-down with t_3 and t_4, while the target is already visible with MDL and t_∞. This explains why the effect sizes are larger for *soccer* than for *arts*.

Figure 6 gives the distribution of the number of drill-down interactions needed to complete one trial, grouped by abstraction technique and target value. In the average target value condition, the distribution of values for MDL is skewed to the right compared to all other approaches; that is, it required fewer drill downs. In the outlier condition, MDL was better than t_3 and t_4, and similar to t_∞.

5.1.4 Discussion

The results confirm that lack of abstraction in views of large hierarchies is detrimental to user performance, at least in visual search tasks. In that respect, even highly abstract approaches, such as t_3 and t_4, are better than unabstracted views. However, as we are not accounting for the latency between drill downs, it is possible that in high latency environments the benefits of abstraction are cancelled by the effect of latency when locating targets requires drill down. In such a case, Figure 6 suggests that MDL would require fewer drill-downs than t_3 and t_4. The fact that a reduction in dataset size was detrimental to user performance in all abstraction conditions suggests that abstraction for small datasets is overkill; nevertheless, compared to t_3 and t_4, MDL had the smallest increase in response time upon dataset reduction.

We expected outlier target to be easier to spot, as their size is ten times larger than the average. The fact that MDL benefits the most of the outlier condition is likely a result of the MDL tendency to expose nodes with large model residuals.

Overall, the average response time of MDL was very similar to that of depth threshold approaches, even though the average clutter in MDL views tends to be higher. In the next section, we investigate the behavior of MDL views using an analytical measure of clutter.

5.2 Feature Congestion Measure

To complement the analysis of the previous section, we compared the same abstraction techniques with the feature congestion measure of clutter [25], which is based on the notion that clutter in a display is associated with degrading performance in visual search. It essentially measures the difficulty of adding a new, salient item to a display. The measure computes the local variability of color and contrast luminance at multiple scales, then combines the values over space and scale to generate a scalar.

We generated treemap views of the DMOZ's subtree "Top/World", the same used throughout this paper, in resolutions ranging from 100x100px to 2400x2400px, with the four abstraction approaches

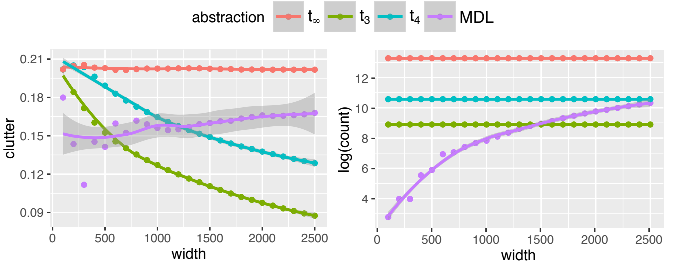


Fig. 7. Left: Level of clutter computed with the Feature Congestion measure as a function of display size. Right: Node count as a function of display size. In both charts, dataset is DMOZ, and aspect ratio is 1:1.

tested in our user study: t_∞, t_3, t_4, and MDL. We then calculated the featured congestion measure using only contrast luminance, as our treemaps do not vary color. In addition, the views were generated without labels, in order to focus on clutter caused by tree structure. Padding is kept, as it is usually necessary for understanding structure in treemaps featuring deep levels.

The results of the experiment are shown in Figure 7, left. The clutter of t_∞ views remains constant and high across the whole range of resolutions. t_3 and t_4 decrease exponentially as the resolution grows. Just like t_∞, the clutter in MDL views remains constant, but is lower than t_∞. Note that for small displays, MDL ends up "falling back" to t_3 and t_4. As space becomes available, the distance between MDL and the depth thresholded views becomes higher, with MDL filling the space with more data.

While clutter is often considered to be unwanted, it is positively correlated with information density, and our approach attempts to find a balance. So, while the clutter of t_3 and t_4 drops dramatically at large screen sizes, so does the information density. Clutter can be compared with the number of nodes visible in the visualizations as seen in Figure 7, right. MDL, t_3, and t_4 consistently reveal far fewer nodes than t_∞. The number of nodes revealed by MDL increases with screen size, while maintaining a roughly constant level of clutter. We argue that while MDL reveals a smaller number of nodes at screen width 1024px, compared to t_3, and the clutter is higher, this is due to the better (more uniform) distribution of nodes across the space, as seen in Figure 8.

Woodruff et al. [31] argue in favor of constant information density (e.g., constant number of objects per area) across x, y and z dimensions of a multiscale visualization. They achieve that automatically by modifying the visual representation of data points and by adjusting the level of abstraction unevenly. Figure 8 (middle) demonstrates that MDL can also approximate constant information density across x and y dimensions. In addition, the results of the feature congestion experiment suggest that MDL approximates constant information density across display resolutions. Across the z dimension (drill-down) we saw variations in the information density caused by the expansion of nodes that have many children (*fan out effect* [4]). Unlike VIDA [31] and GrouseFlocks [4], which create new aggregates to minimize fan out, our method preserves the original hierarchy structure. As a result, if there are strong variations in how wide the first levels of subtrees are, the information density can vary.

5.3 Scalability Analysis

Our algorithm is a customization and application of the approach of Li and Abe, with the additional optimization step of including a factor of display size. They find that determining the MDL tree cut terminates in time $\mathcal{O}(NxS)$, where N denotes the number of leaf nodes in the input tree T and S denotes the input sample size. Our algorithm increases this procedure by the transformation from the data domain to the pixel domain, and the estimation of the probabilities of leaves, both of which are $\mathcal{O}(NxS)$. As S is generally much larger than N, our algorithm scales roughly linearly with the size of the dataset.

Any overhead encountered by generating a display-optimized tree cut could be shifted to a server-side pre-calculation, for example, to

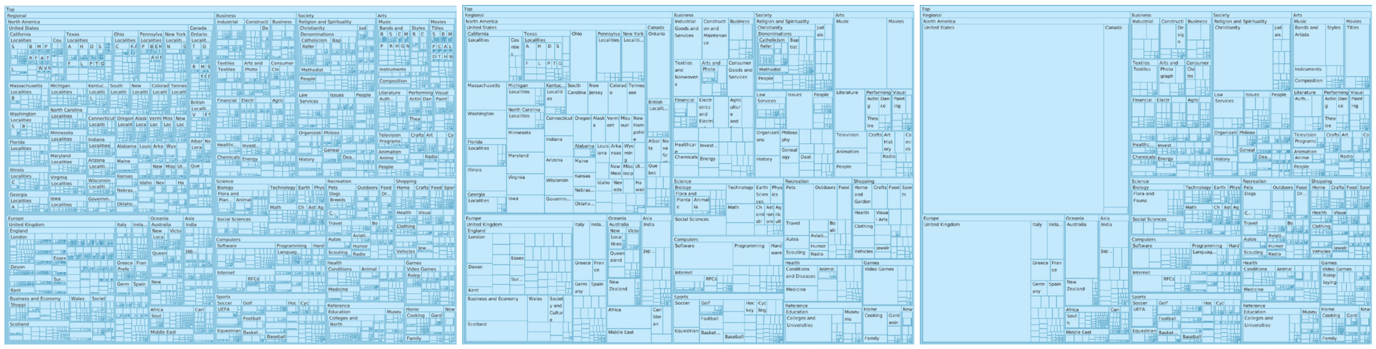


Fig. 8. Treemap views of the DMOZ dataset for a 1024x768px display. On the left, the full view of the dataset. In the middle, the level of detail is based on the best MDL tree cut (uneven). On the right, an even cut is performed below level 4 of the tree. The MDL tree cuts yield a better balance of information density and clutter.

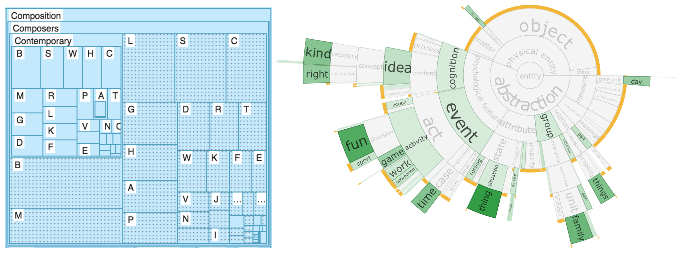


Fig. 9. Subtle visual cues for collapsed nodes using texture (treemap) and border thickness (sunburst).

pre-cache the tree cut for a variety of standard screen sizes, thereby eliminating any delay incurred by the tree cut operation. The resulting trees generally better balance information density with the number of nodes, and will render faster and consume fewer client resources than an equivalent full tree, and show a more uniform information density than a fixed-level tree cut.

6 Discussion

Generality: With the formulae for treemap and sunburst visualizations, we exemplified how model selection criteria can be written for visualizations under the MDL framework. We believe that MDL can be applied to many other kinds of hierarchical visualizations where (a) some visual aspect of the nodes is weighted by a score, and (b) the scores are cumulative. This might include visualizations that are not traditionally hierarchical but were augmented with multiscale functionality, such as aggregated scatterplots, parallel coordinates and node-link diagrams [11]. Defining criteria for new classes of visualization involves the specification of three main expressions:

1. The transformation from the data domain to the visualization domain (pixel units) ($A(C)$).
2. Number of bits necessary to encode the visualization ($L(V)$).
3. How probabilities of tree leaves are estimated from the visual representation of classes ($P(l)$).

Uniform distribution assumption: Behind the estimation of the probability of leaves given a class is the assumption of uniform probability. If this assumption is not reasonable in a certain application domain, $P(l)$ can be easily changed to reflect a different probability distribution. A case where this might be useful is when depicting geographic information, where the user might have a prior assumption about the distribution; for example, given a certain value for the State of New York (e.g., gross product), one might expect that value to be concentrated in New York City. In many other cases, lacking prior knowledge, we expect the uniform distribution to be fairly reasonable.

An important limitation is that if the data is uniformly distributed, the tree cut generated will be the most abstract possible (i.e. the root). This occurs, for example, if the value of every leaf is 1. Likewise, if a different distribution is used and the data conforms exactly, the tree cut will be overgeneralized.

Interpretation: It is especially important in models such as ours, where abstraction is calculated algorithmically, that the presence of data abstraction is made apparent, in ways that are not distracting to the main task of working with a visualization. While our technique and evaluation focus on the level of abstraction, we have begun an investigation into the representation problem. Figure 9 suggests preliminary visual designs which subtly distinguish aggregates from regular leaves. On treemaps, aggregates are textured; on sunbursts, collapsed nodes are decorated with a colored, thicker border.

7 Conclusion

In this paper we have presented a technique for using the MDL Principle, extended with considerations of display space, in order to create optimized views of hierarchical datasets which fit the “analyze first, show the important” first step of the visual analytics pipeline. In addition to providing overviews customized to dataset and display size characteristics, the display-optimized tree cuts can be interactively expanded by changing the weighting parameters.

The number of nodes displayed in a display-optimized MDL tree cut is similar to those in an even tree cut at a set depth, but fewer than showing a full tree. This increases the rendering efficiency, resulting in a performance gain in web-based visualization applications, where processing resources, memory, and display space may be constrained (e.g. on mobile devices). In addition, on small screens and any touch device where rendered elements are small, interaction accuracy can be difficult due to the “fat fingers” problem. Our technique applies abstraction in cluttered areas of a visualization, which will likely improve target selection accuracy.

We have demonstrated our technique applied to two datasets across two different hierarchical visualization types, treemap and sunburst diagrams, and outlined the steps required to generalize the approach to other visualization types. Display-optimized MDL tree cuts may prove especially useful due to their general nature — they are not customized to dataset characteristics. However, it is also possible to tailor them to the dataset, for example, by basing the tree cut on a selected data attribute, as long as that attribute is quantitative on the leaves and cumulative in the hierarchy.

In the future, we plan to apply the display-optimized MDL tree cut to new visualization types. In addition, we see promise in the challenge of developing new methods for representing abstraction. Finally, while we demonstrate the possibilities of interactive drill down to deeper levels of the tree cut using a fixed step size, we plan to investigate ways to automatically tailor the drill down step based on dataset characteristics, display space, and to harmonize tree cut drill down with more traditional techniques to click and open branches manually.

References

[1] J. Abello, F. Van Ham, and N. Krishnan. ASK-GraphView: A large scale graph visualization system. *IEEE Trans. on Visualization and Computer Graphics*, 12(5):669–676, 2006.

[2] N. Andrienko and G. Andrienko. Spatial generalisation and aggregation of massive movement data. *IEEE Trans. on Visualization and*, 17(2):205–19, 2010.

[3] D. Archambault and D. Greene. ThemeCrowds: Multiresolution summaries of twitter usage. *International Workshop on Search and Mining User-generated Contents*, pages 77–84, 2011.

[4] D. Archambault, T. Munzner, and D. Auber. GrouseFlocks: Steerable exploration of graph hierarchy space. *IEEE Trans. on Visualization and Computer Graphics*, 14(4):900–913, 2008.

[5] S. Baldassi, N. Megna, and D. C. Burr. Visual Clutter Causes High-Magnitude Errors. *PLoS Biology*, 4(3):e56, 2006.

[6] A. Brew, D. Greene, D. Archambault, and P. Cunningham. Deriving insights from national happiness indices. *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 53–60, 2011.

[7] M. Chuah. Dynamic aggregation with circular visual designs. In *Proceedings IEEE Symposium on Information Visualization*, pages 1–9, 1998.

[8] C. Collins, S. Carpendale, and G. Penn. DocuBurst: Visualizing document content using language structure. *Computer Graphics Forum*, 28(3):1039–1046, jun 2009.

[9] Q. Cui, M. Ward, E. Rundensteiner, and J. Yang. Measuring data abstraction quality in multiresolution visualizations. *IEEE Trans. on Visualization and Computer Graphics*, 12(5):709–716, 2006.

[10] G. Ellis and A. Dix. A taxonomy of clutter reduction for information visualisation. *IEEE Trans. on Visualization and Computer Graphics*, 13(6):1216–1223, 2007.

[11] N. Elmqvist and J.-D. Fekete. Hierarchical aggregation for information visualization: overview, techniques, and design guidelines. *IEEE Trans. on Visualization and Computer Graphics*, 16(3):439–54, 2010.

[12] S. Haroz and D. Whitney. How Capacity Limits of Attention Influence Information Visualization Effectiveness. *IEEE Trans. on Visualization and Computer Graphics*, 18(12):2402–2410, dec 2012.

[13] D. Keim, F. Mansmann, J. Schneidewind, and H. Ziegler. Challenges in Visual Data Analysis. In *Proc. of the Int. Conv. on Information Visualisation*, pages 9–16. IEEE, 2006.

[14] D. Koutra, U. Kang, J. Vreeken, and C. Faloutsos. Summarizing and Understanding Large Graphs. *Statistical Analysis and Data Mining*, 8(3):183–202, 2015.

[15] R. Lamarche-perrin, Y. Demazeau, and J.-m. Vincent. Building Optimal Macroscopic Representations of Complex Multi-agent Systems. In N.-T. Nguyen, R. Kowalczyk, J. Corchado, and J. Bajo, editors, *Trans. on Computational Collective Intelligence XV*, volume 8670, pages 1–27. Springer Berlin Heidelberg, 2014.

[16] R. Lamarche-perrin, L. M. Shnorr, J.-m. Vincent, and Y. Demazeau. Evaluating Trace Aggregation Through Entropy Measures for Optimal Performance Visualization of Large Distributed Systems. Technical report, INRIA, 2012.

[17] T. C. M. Lee. A Minimum Description Length Based Image Segmentation Procedure , and Its Comparison with a Cross-Validation-Based Segmentation Procedure. *Journal of the American Statistical Association*, 95(1995):259–270, 1999.

[18] T. C. M. Lee. An Introduction to Coding Theory and the Two-Part Minimum Description Length Principle. *International Statistical Review*, 69(2):169–183, 2001.

[19] H. Li and N. Abe. Generalizing Case Frames Using a Thesaurus and the MDL Principle. *Computational linguistics*, 24(2):217–244, 1998.

[20] G. a. Miller. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological review*, 101(2):343–352, 1956.

[21] T. Munzner. *Visualization Analysis & Design*. CRC Press, 2014.

[22] J. R. Quinlan and R. Rivest. Inferring Decision Trees Using the Minimum Description Length Principle. *Information and Computation*, 80(1989):227–248, 1989.

[23] J. Rissanen. A universal prior for integers and estimation by minimum description length. *Annals of Statistics*, 11(2):416–431, 1983.

[24] J. Rissanen. *Stochastic Complexity in Statistical Inquiry*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1989.

[25] R. Rosenholtz, Y. Li, Z. Jin, and J. Mansfield. Feature congestion: A measure of visual clutter. *Journal of Vision*, 6(6):827–827, 2010.

[26] R. Rosenholtz, Y. Li, and L. Nakano. Measuring visual clutter. *Journal of vision*, 7(2):17.1–22, 2007.

[27] B. Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. In *Proc. 1996 IEEE Symposium on Visual Languages*, pages 336–343. IEEE Comput. Soc. Press, 1996.

[28] F. Töpfer and W. Pillewizer. The Principles of Selection. *The Cartographic Journal*, 3(1):10–16, 1966.

[29] A. Wagner. Enriching a lexical semantic net with selectional preferences by means of statistical corpus analysis. In *Proceedings of the European Conference on Artificial Intelligence - ECAI*, 2000.

[30] J. M. Wolfe. Visual Search. *Attention*, pages 1–41, 1998.

[31] A. Woodruff, J. Landay, and M. Stonebraker. Constant density visualizations of non-uniform distributions of data. In *Proceedings of the 11th annual ACM symposium on User interface software and technology - UIST '98*, pages 19–28, 1998.

[32] Y. Wu, X. Liu, S. Liu, and K. L. Ma. ViSizer: A visualization resizing framework. *IEEE Trans. on Visualization and Computer Graphics*, 19(2):278–290, 2013.

[33] B. Yost and C. North. The perceptual scalability of visualization. *IEEE Trans. on Visualization and Computer Graphics*, 12(5):837–844, 2006.