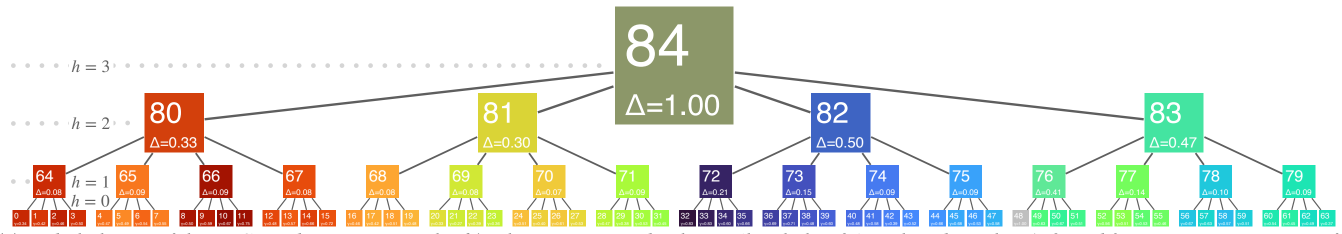
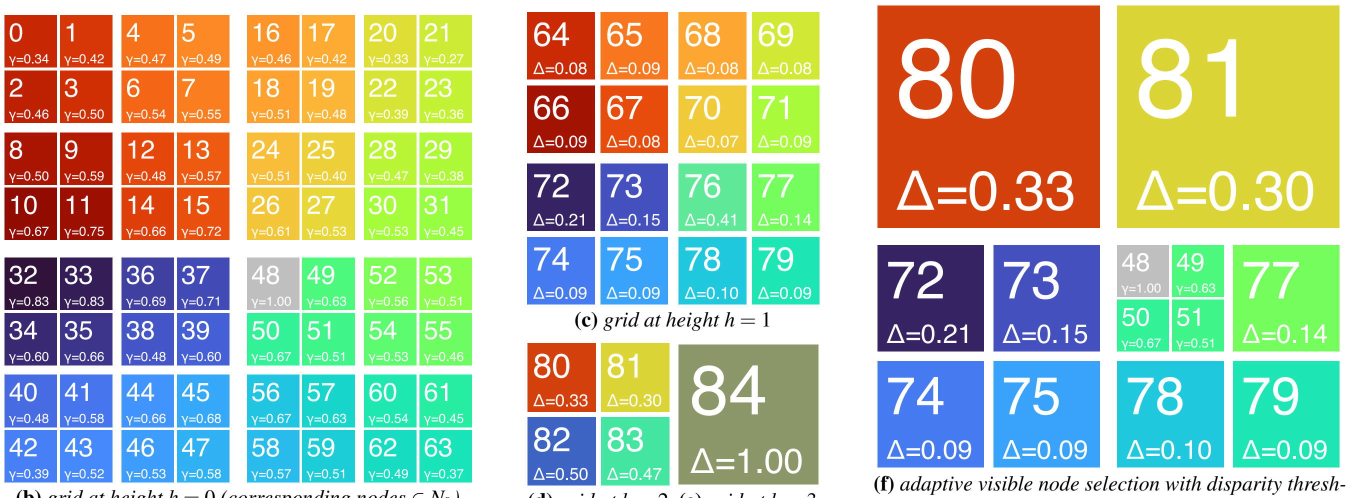
Optimizing Grid Layouts for Level-of-Detail Exploration of Large Data Collections – Supplementary Material

S. Frey

University of Groningen, the Netherlands



(a) *node-link view of the LDG quadtree structure; the 64 colors are assigned to leaves (height $h = 0$), with nodes at $h = 1$, 2, and 3 containing aggregates of their respective leaves*



(b) *grid at height $h = 0$ (corresponding nodes $\in N_0$)* **(c)** *grid at height $h = 1$* **(d)** *grid at $h = 2$* **(e)** *grid at $h = 3$* **(f)** *adaptive visible node selection with disparity threshold $\tau = 0.4$*

Figure 1: *LDG for 63 members of the Turbo colormap and one grayscale color in (a) node-link view and (b–e) grid view for different visible nodes. Each node (and corresponding grid area) depicts the (average) color of its member(s), node id (large, top), and normalized evaluation value γ ($h = 0$) or disparity Δ ($h > 0$). (f) adaptive visibility selection via τ.*

Abstract

This document provides additional material for the EuroVis 2022 paper with the title "Optimizing Grid Layouts for Level-of-Detail Exploration of Large Data Collections". In particular, it provides more in-depth information regarding the presentation of LDGs, as well as further evaluations.

Table 1: *Notation used for the description of LDGs*

N	nodes in the LDG representation
n_{root}	root node ($\in N$)
h	height of nodes N (i.e., distance to the root node n_{root})
N_h	set of nodes $N_h \subset N$ at height h (e.g., N_0: leaf nodes at $h = 0$)
$N_{\uparrow}(n)$	gives all ancestors of n (i.e., all nodes on the path to the root n_{root})
$N_0(n)$	leaf nodes of the corresponding subtree of $n \in N$: $\{n_0 \in N_0 \mid n \in N_{\uparrow}(n_0)\}$
T	set of members in a data collection
$d : T \times T \to \mathbb{R}$	distance between members $\in T$
$A : N_0 \to T$	assignment of leaves (grid cells) to collection members
$\bar{t}(n)$	the aggregate of all members assigned to corresponding leaves $N_0(n)$ of a node n
$\Gamma : A \to \mathbb{R}$	objective function quantifying the cost of assignment A
$\dot{\gamma} : N \times T \times A \to \mathbb{R}$	cost of placing a member $\in T$ at a node $\in N$ under grid assignment A, only considering the hierarchy
$\gamma : N \times T \times A \to \mathbb{R}$	extension of $\dot{\gamma}$ that considers neighborhood as well
$\Delta(n)$	disparity: a normalized measure of how well aggregate representation $\bar{t}(n)$ captures its members
τ	disparity threshold for adaptive selection of visible nodes

1. LDG Notation

Table 1 provides an overview on the notation used for the description of LDGs.

2. LDG Presentation

Grid View

When presenting LDGs, their structure is reflected by adjusting the spacing between node tiles accordingly (Fig. 1). From grid cell coordinates g along any axis, gaps to the top or left are determined based on the distance to the closest common parent regarding the cell to the top or left, respectively:

g	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\sigma(\cdot)$	–	1	2	1	3	1	2	1	4	1	2	1	3	1	2	1

Naturally, there is no spacing for the leftmost (or the topmost) node. The position of every cell in the grid is then computed via $g + \omega \sum_{i \in \{1 \dots g\}} \sigma(i)$, with ω being a scaling factor controlling the width of the spacing. Inner nodes $N_{h>0}$ use the grid coordinates of their top-left-most node $\in N_0$.

Tiles of inner nodes (i) are sized to cover a range of grid cells, and (ii) are further scaled to depict the ratio of regular to void members (e.g. Fig. 2b). The tile's side length of node n at height h is as follows:

$$\underbrace{\left(2^h + \omega \sum_{i \in \{1,\dots,2^h-1\}} \sigma(i)\right)}_{\text{(i) reserved tile area in the grid}} \quad \underbrace{\sqrt{\frac{|N_0(n)|}{4^h}}}_{\text{(ii) scaling with regular member ratio}} \quad . \qquad (1)$$

Tiles are further centered within their cell at the barycenter of their corresponding non-void tiles to reflect their respective positions in the grid (Fig. 2).

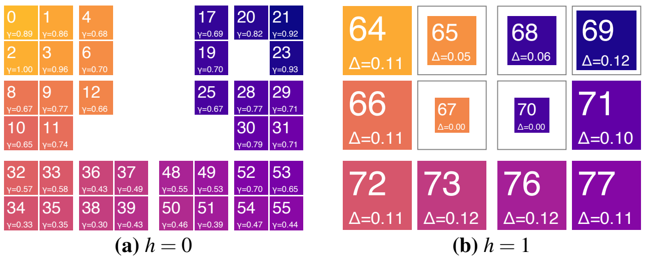


Figure 2: 8×6 *LDG with 37 colors of the Plasma colormap and 11 void members. (a) On $h = 0$, void members help to further emphasize similarity relationships by forming void spaces. (b) On $h \geq 1$, the tile representation is scaled with the ratio of regular to void members.*

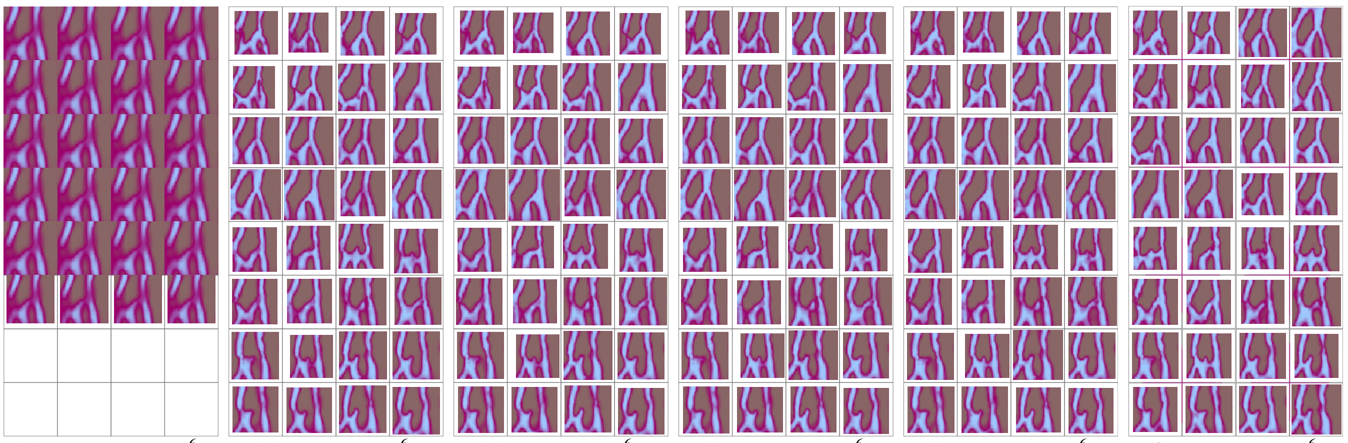
Exploration

Although the focus of this work is on the LDG representation, suitable exploration modalities are naturally required for data analysis in practice. In our current prototype, we provide different means for interaction, which—besides panning and zooming the presentation—allow to adjust the set of visible nodes in three different ways: globally via (i) height h or (ii) disparity Δ, and (iii) through direct selection of tiles for expansion or collapse. Directly interacting with a tile at height h allows to expand respective nodes to $h - 1$ or collapse them to $h + 1$. We optionally also consider the tiles within the 8-neighborhood to reflect that similar members also of interest may be placed in the vicinity with our LDGs. Hovering a tile with the mouse highlights the considered neighborhood. In general, tiles fade in as they become visible to outline the changes resulting from user interaction.

3. Performance Analysis and Comparison

Prototype Viewer

On `https://ldg-demo.github.io` we provide a prototype viewer where all layouts of discussed use cases can be explored interactively. It is a port of the desktop application written in C++ using the widget toolkit Qt based on JavaScript and WebAssembly. Please note that while it cannot provide the performance of desktop tool and exhibits other limitations, we still deemed it to be beneficial to provide it as it allows to explore the LDGs beyond what is demonstrated in the paper. Among others, we could confirm that it works well with Safari, Chrome and Firefox. Compromises regarding tile image resolutions were necessary to keep data sizes as low as possible in the web environment. Still, user interactions like lowering the disparity threshold or choosing a low level can take a while to take effect if a lot of tiles have to be loaded for the first



(a) *0* ($\Gamma = 3.81\times10^6$) **(b)** *32* (2.87×10^6) **(c)** *64* (2.77×10^6) **(d)** *128* (2.66×10^6) **(e)** *256* (2.54×10^6) **(f)** *3520* (2.20×10^6)

Figure 3: *Refinement results at $h = 6$ for an MCMC run, demonstrating quick convergence on higher levels. A modified color map emphasizes ambiguity (pink replaces white).*

time (depending on the machine and the network connection); we ask for patience in such cases. In general, at least the reduced cases for comparison should be quick to load and interact across a wide range of connection speeds and processors.

The performance of our LDG optimization approach is investigated by means of sixteen individual runs conducted for (variants of) each use case with an AMD Ryzen 7 2700X Eight-Core Processor and 32 GB RAM. Results at different stages of refinement for one run are shown in Fig. 3. The initial random assignment yields similar superimposition results across nodes with high uncertainty ((a), with cells in bottom rows only featuring void members). 32 passes already significantly reduce disparity and clear structures emerge in tiles (b). Additional passes refine this further (e, f), but the summary presented at $h = 6$ in (f) remains largely stable as predominantly fine-granular, local adaptations occur.

An important property is stability, and for this the results of sixteen independent runs are compared in Fig. 4: while there naturally is some variation, the basic structures are very similar and demonstrate no significant impact of stochastic factors in the result. This is also reflected in the respective Γ-values as can be seen from the respective line chart in the main paper.

LDG is now compared against two state-of-the-art distance-preserving grid techniques—Kernelized Sorting [QSST10] and IsoMatch [FDH*15]—as well as a hierarchical extension to IsoMatch. Caltech and Stock with a subset of $|T| = 1024$ members is used as both aforementioned techniques exhibit approximately cubic complexity regarding $|T|$ and computationally cannot efficiently handle larger ensembles (see performance discussion below).

We consider disparity to quantitatively assess how well aggregate representations at a node are suited to convey their members. Fig. 5 shows that LDG yields significantly lower disparity across all levels, confirming that branches are more homogeneous (as could be expected). IsoMatchHG significantly improves over IsoMatch's disparity, underlining what has been found above through visual inspection. Comparing the two use cases, Caltech yields much higher node disparities than Stock, which is due to both the more diverse

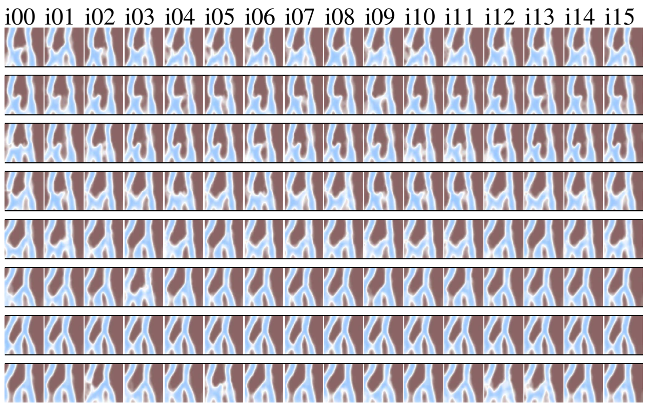


Figure 4: *Aggregate representations of sixteen individual runs i00–i15 at height $h = 7$. Results are similar throughout despite using different seeds.*

ensemble and the richer feature representation (2048 versus 23 entries). However, there are subsets of both ensembles with different properties. In Caltech, there are outliers toward lower scores, indicating larger homogeneous groups (most notably, faces, airplanes, and motorbikes). The opposite is true for Stock: here the outliers showing higher disparity reflect the extreme cases of falling and rising prices.

While the primary focus of this work is to achieve visual scalability for large ensembles, computational scalability is naturally a necessary requirement. For Isomatch and Kernelized Sorting we used the default implementations provided by the authors, and neither is able to generate results beyond thousands of members and grid cells due to strongly increasing runtimes (Fig. 6). For hierarchical clustering, we employed the implementation of agglomerative clustering with default parameters (i.e., using Ward linkage) provided by scikit-learn [PVG*11] (0.24.2). Runtimes are comparably fast, although also exhibiting polynomial scaling: 11 thousand members of Stock$\rightarrow$2.8 s, 22 k$\rightarrow$12.6 s, 45 k$\rightarrow$65.2 s). Unfortunately, we were not able to produce results beyond $\approx 45\,000$ members due to

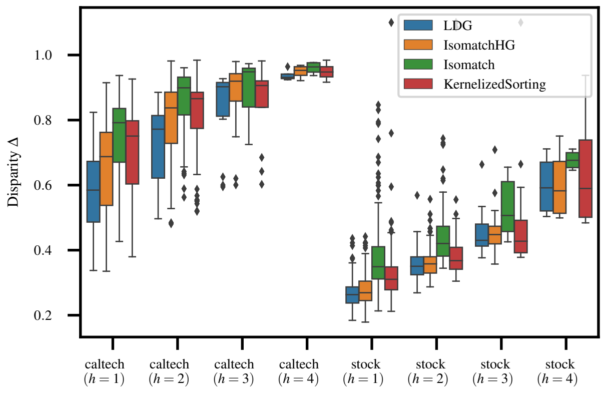


Figure 5: *Disparity distribution of nodes at different levels across methods (lower is better).*

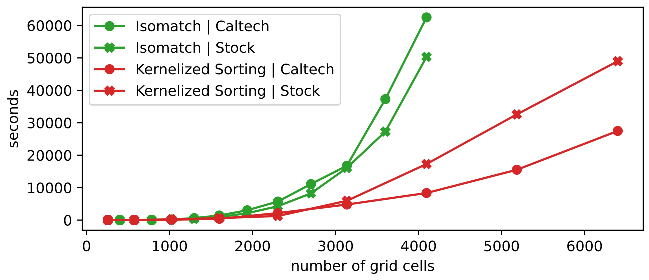


Figure 6: *Both Isomatch and Kernelized Scaling exhibit polynomial scaling with the number grid cells (using a random subset of members of the same size in this experiment).*

prohibitively high memory requirements. This issue could potentially be addressed by one of the numerous variants with different properties and recent work focusing on scalability (e.g. [MKK*19]).

References

[FDH*15] Fried O., DiVerdi S., Halber M., Sizikova E., Finkelstein A.: IsoMatch: Creating informative grid layouts. *Computer Graphics Forum 34*, 2 (2015), 155–166. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12549, doi:10.1111/cgf.12549. 3

[MKK*19] Monath N., Kobren A., Krishnamurthy A., Glass M. R., McCallum A.: Scalable hierarchical clustering with tree grafting. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (New York, NY, USA, 2019), KDD '19, ACM, pp. 1438–1448. doi:10.1145/3292500.3330929. 4

[PVG*11] Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., Blondel M., Prettenhofer P., Weiss R., Dubourg V., Vanderplas J., Passos A., Cournapeau D., Brucher M., Perrot M., Duchesnay E.: Scikit-learn: Machine Learning in Python. *The Journal of Machine Learning Research 12* (2011), 2825–2830. 3

[QSST10] Quadrianto N., Smola A. J., Song L., Tuytelaars T.: Kernelized sorting. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32*, 10 (Oct 2010), 1809–1821. doi:10.1109/TPAMI.2009.184. 3