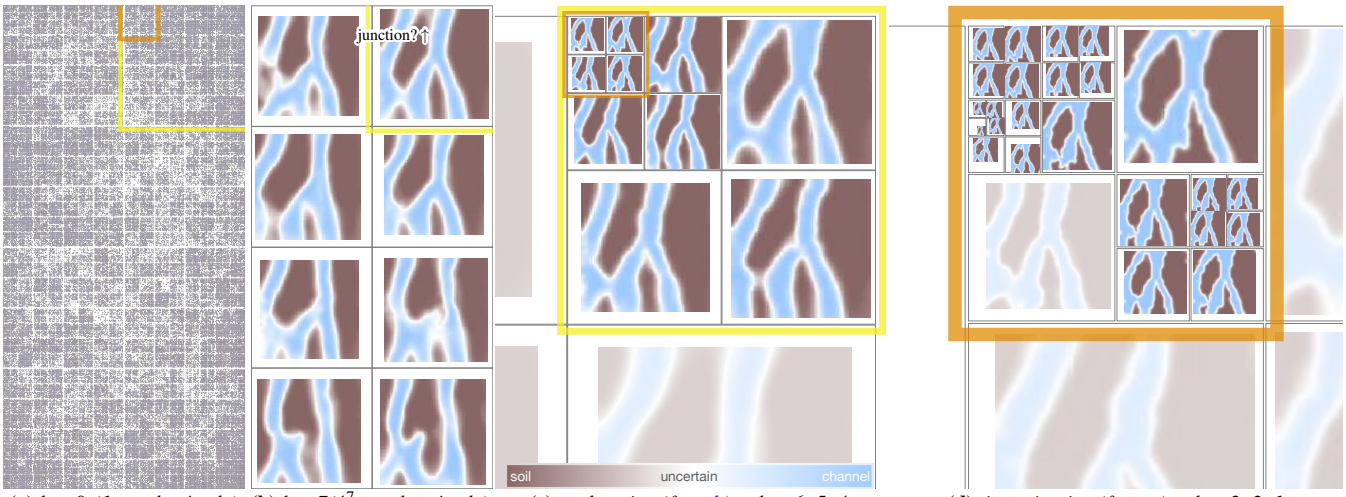
DOI: 10.1111/cgf.14537
Eurographics Conference on Visualization (EuroVis) 2022
R. Borgo, G. E. Marai, and T. Schreck
(Guest Editors)

Volume 41 (2022), Number 3

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

S. Frey

University of Groningen, the Netherlands



(a) $h = 0$ *(1 member/node)* **(b)** $h = 7 (4^7$ *members/node)* **(c)** *exploration (from b) to* $h = 6, 5, 4$ **(d)** *investigation (from c) to* $h = 3, 2, 1$

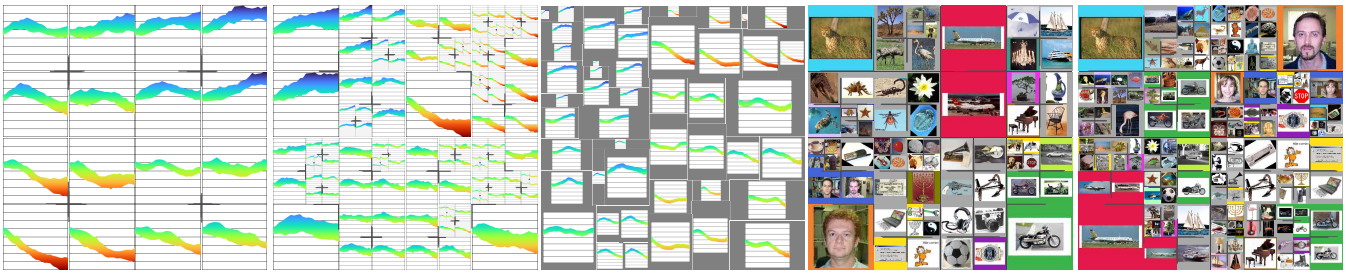
Figure 1: *Level-of-detail grid (LDG) for* 95 000 *channel structures in soil from a probability distribution (MCMC). (a) At the highest granularity level (height* $h = 0$*), each member is individually depicted, resulting in too many and too small tiles for expressive analysis (similar to standard "flat" grid layouts). (b) LDGs allow to start from a visual summary at low granularity (height* $h = 7$*, where each tile represents* $4^7 = 16384$ *members). Here, among others, this hints at the existence of junctions between channels at the top of the domain. (c) This is investigated more closely via node expansion to higher detail (*$h = 6 \rightarrow 5 \rightarrow 4$*), confirming the occurrence of such junctions (a zoom-in is shown, LDGs always consistently represent the full data). (d) For these cases, exploration to lower granularity (*$h = 3 \rightarrow 2 \rightarrow 1$*) reveals exactly one additional junction between channels in the lower third and three channel entries at the bottom of the domain.*

Abstract

This paper introduces an optimization approach for generating grid layouts from large data collections such that they are amenable to level-of-detail presentation and exploration. Classic (flat) grid layouts visually do not scale to large collections, yielding overwhelming numbers of tiny member representations. The proposed local search-based progressive optimization scheme generates hierarchical grids: leaves correspond to one grid cell and represent one member, while inner nodes cover a quadratic range of cells and convey an aggregate of contained members. The scheme is solely based on pairwise distances and jointly optimizes for homogeneity within inner nodes and across grid neighbors. The generated grids allow to present and flexibly explore the whole data collection with arbitrary local granularity. Diverse use cases featuring large data collections exemplify the application: stock market predictions from a Black-Scholes model, channel structures in soil from Markov chain Monte Carlo, and image collections with feature vectors from neural network classification models. The paper presents feedback by a domain scientist, compares against previous approaches, and demonstrates visual and computational scalability to a million members, surpassing classic grid layout techniques by orders of magnitude.

1. Introduction

Advances in simulation methods and image acquisition technology allow to acquire vast collections of associated data points—hereafter also referred to as ensembles—which support a comprehensive analysis of underlying processes [WHLS19]. However, obtaining an overview of large ensembles beyond tens of thousands of members is



(a) *LDG,* $\tau = 0.4$*, Stock* **(b)** *IsomatchHG,* $\tau = 0.4$ **(c)** *Hierarchical Clustering* **(d)** *LDG,* $\tau = 0.85$*, Caltech* **(e)** *IsoMatchHG,* $\tau = 0.85$

Figure 2: *Representations of 1024 members from (a–c) a stock prediction ensemble and (d,e) an image collection. (a,d) Our LDG optimizes for low variance both within the hierarchy and across grid neighbors. To the best of our knowledge, there is no directly comparably method, yet we adapted the hierarchical extension to Isomatch for the sake of comparison [FDH*15] (IsomatchHG). (b,e) IsomatchHG iteratively splits the grid and with it the ensemble members into equally-sized partitions yielding screen-space efficient grid like LDGs, but cannot prevent strong discontinuities across grid neighbors (e.g., placing members with strong downward trend directly next to those with positive development in the grid center in (b)). It also generally achieves less homogeneity within clusters, forcing a much higher granularity for the same disparity (uncertainty) threshold* τ *, and scatters similar members across the grid (e.g., motorbikes in (e)). (c) Hierarchical clustering (Ward linking) minimizes the variance of the clusters, but does not consider (treemap) placement, resulting in spatial discontinuities and separated members.*

difficult, especially when there are no associated hierarchies, facets, parameters, etc. to provide some explicit structure. Many scientific ensembles further exhibit smooth distributions of members which means that they are difficult to expressively classify or cluster into distinct groups. This is often the case with simulation ensembles—even if some cases with characteristic behavior or structures may be identified by an expert, a range of results typically falls between such distinctive categories, like the channel structures in Fig. 1 obtained via Markov-Chain Monte Carlo (MCMC) [RXN20] or stock developments in Fig. 2a–c from a Black-Scholes Model [BS73]. This paper proposes a computationally scalable approach for creating level-of-detail grids to also achieve visually scalability for large ensembles without associated structure or distinct partitions, solely based on distances between members.

Common state-of-the-art distance-preserving grid layouts [FDH*15, QSST10, SG14] present members in the cells of a grid such that similar members are close. They are well-suited for adequately conveying continuously changing members while making efficient usage of screen space. However, showing visual representations for ensembles with hundreds of members and more does not scale visually due to too many tile images shown in tiny cells (e.g., Fig. 1a). Hierarchical extensions have the potential to address visual scalability at least, but previous methods toward this yield grids with strong discontinuities and separated similar members (Fig. 2b and Fig. 2e). Alternatively, hierarchical clustering methods are able to generate homogeneous partitions of the provided data, but corresponding treemap representations—apart from being not as screen space-efficient as grid-based techniques—cannot adequately reflect similarities across clusters (Fig. 2c). This is particularly problematic for ensembles whose members cannot clearly be separated into distinct classes.

In this work, we introduce level-of-detail grid layouts (LDG) along with an efficient optimization approach to generate them for large ensembles. LDGs combine the screen space-efficiency and high neighborhood similarity of traditional distance-preserving grids with the visual scalability of hierarchical approaches (Fig. 1, Fig. 2a). (2D) grids are organized in a quadtree: leaves (height $h = 0$) correspond to one grid cell—each featuring one assigned member—while inner nodes ($h > 0$) cover a quadratic range of cells ($2^h \times 2^h$). Inner nodes present the corresponding 4^h members from the respective leaves aggregated in one image tile, also directly conveying involved disparity, if possible (e.g., the variation of channel structures in Fig. 1 or the range of stock prices in Fig. 2). LDGs can convey an ensemble overview with nodes at uniform heights—whereas each tile represents about the same number of members (Fig. 1b)— or via adaptive refinement in which grid areas with higher disparity are depicted with finer granularity (e.g., Fig. 2d). They also support arbitrary local refinement by a user during exploration (e.g., Fig. 1b–d).

LDGs do *not* aim to partition the data into distinct clusters but use a hierarchical grid structure for level-of-detail presentation. LDGs are optimized for homogeneity (minimizing disparity) within inner nodes and across grid neighbors when assigning members to cells—based on pairwise distances between members only. Our optimization approach follows a progressive approach, iteratively refining the LDG and quickly yielding expressive results. The parallel grid optimization method employs a hierarchical scheme to achieve computational scaling to large ensembles; we demonstrate this for up to a million members in this paper.

The main contribution of this work is the introduction of LDGs along with a computationally scalable optimization approach to generate the representation from large ensembles. To the best of our knowledge, LDGs are able to convey substantially larger ensembles than previous grid layout techniques. Below, the paper reviews related work (Sec. 2), introduces LDGs (Sec. 3), and describes the optimization approach to generate them (Sec. 4). Utility is demonstrated via diverse use cases (Sec. 5), performance evaluation (Sec. 6), and further comparison against prior approaches (Sec. 7). The paper concludes with a discussion and outline of future work (Sec. 8).

2. Related Work

Distance-preserving techniques place data points (e.g., ensemble members) as graphical elements such that similarity relationships are reflected. Multidimensional projection techniques like t-

SNE [vH08], LAMP [JCC*11], or UMAP [MHM18] have emerged as fundamental analysis tools, but using information-rich tiles with 2D visualizations generally yields overlap and visual clutter [JCC*11]. While various techniques introduce additional measures to prevent overlaps—e.g., RWordle [SSS*12], IncBoard [PodAL10], ProjSnippet [GRP*14], and UnTangle Maps [CLG16]—they tend to inefficiently use visual space.

Addressing this, other approaches directly assign members to cells under the consideration of similarity relationships. Self-Organizing Maps (SOMs) are unsupervised neural networks creating a discretized lower-dimension grid representation of the data [Koh90]. However, several members are generally mapped to a single grid cell, yielding overlap on the composed grid [FDH*15, QSST10]. Generative Topographic Mapping is a probabilistic approach conceptually related to SOMs also suffering from overlap [BSW98]. Spectral Hashing creates codes such that their Hamming distance approximates member distances [WTF09]. Codes are split into bins to yield a grid mapping, but hash collisions also yield overlaps. Starting from an input projection, NMAP employs a space-filling method to generate regular placements in power-of-two grids via recursive bisection and scaling [DSF*14]. DGrid also employs binary space partitioning in combination with multidimensional projections [HP19]. CorrelatedMultiples employ a constrained multidimensional scaling solver that preserves spatial proximity while forcing items to fit within a fixed region [LHNS18]. Self-Sorting Map uses a permutation procedure to maximize the cross-correlation between member and cell distances by swapping cells in different (sub)quadrants [SG14]. The optimization procedure proposed in this work bears some similarity in that it also partitions members into sub-groups and locally solves optimization problems. Kernelized Sorting solves a quadratic assignment problem from members to grid cells [QSST10]. IsoMatch [FDH*15] projects data to a plane using ISOMAP [TSL00], creates a bipartite graph between projection and grid positions, and finally uses the Hungarian method [Kuh55] for assignment. The Hungarian method is also used to create LDGs, but crucially it is applied to partitions of fixed size instead of the full ensemble, avoiding scalability issues arising from the approximately cubic complexity. We compare against Kernelized Sorting and IsoMatch in Sec. 7 (also see supplemental material).

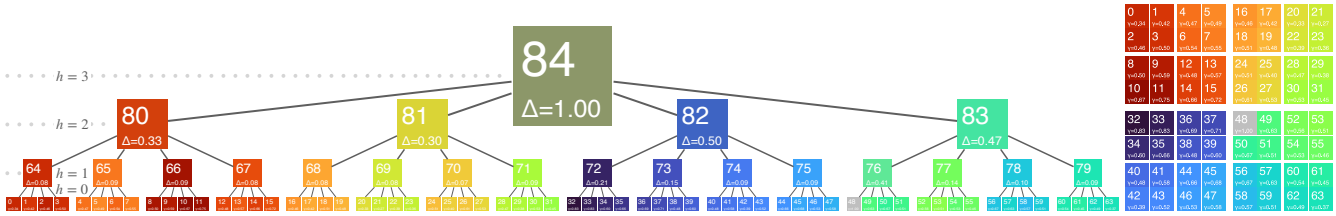
Grid layouts generally suffer from visual scalability problems for large numbers of members: representing each member on its own results in too many and too small tiles to be useful for analysis. Hierarchical techniques can significantly improve scalability, and various hierarchical visualization approaches have been proposed for geographical, hierarchical or multivariate data with different facets [SHS11,SDW09] (e.g., via treemaps [WD08,SMS*20]). However, the data must have respective information associated with it, or it needs to be created prior to the presentation. Hierarchical clustering builds a hierarchy of clusters either bottom-up via merging or top-down via splitting, aiming to minimize the dissimilarity within clusters [Joh67]. However, the partitioning into distinct clusters can be ambiguous for ensembles with non-distinct member groups, and it is unaware of the targeted visual presentation (see Fig. 2c).

Extensions beyond plain 2D grids have been proposed for data with no explicitly associated hierarchy to tackle visual scalability, but they exhibit significant shortcomings regarding the aim of explorable visual summaries. For this, Kernelized Sorting [QKTB10] simply replaces the 2D grid with a 3D pyramidal version, whereas each cell of the pyramid has one member assigned to it at any height. This does not yield a hierarchy, and the grid cells toward the top of the pyramid do not summarize members below. Likewise, some techniques like those based on SOMs allow to choose smaller grids and assign multiple members per cell, yet the partitioning may be highly imbalanced and there is no consistent global representation on the level of individual members. An extension to IsoMatch [FDH*15] generates a hierarchical structure top-down by iteratively clustering ensemble subsets, exemplified in the paper for 4096 members represented by 4×4-grids. Each cell contains $4096/16 = 256$ members in one top-level grid (choosing one image as representative per cell). For each of the 16 top-level grid cells another 4×4 grid is created (here, one cell contains 16 members), for which then again individual 4×4-grids are produced with one member per cell. Critically, each of these $1 + 16 + 256 = 273$ grids is generated and shown independently from the others, and there is no consistent layout across nodes. The full ensemble is only summarized at the 4×4 top-level grid, with no flexibility of changing the granularity of the overview during exploration. Selecting any node for a closer view only conveys these corresponding members without embedding into a larger context. In contrast, the aim of this paper is a consistent global representation with high similarity within hierarchy branches and across neighbors in the grid. A comparison to a further extended version of this original approach toward compatible grid layouts called IsomatchHG is provided in Fig. 2 and Sec. 7.

Other related approaches include a SOM-based method for interactive browsing that organizes image collections as a connected graph with edges between similar images [BHM16]. Pan et al. [PTD*21] generate visually pleasing layouts from image collections based on design principles and cognitive psychology, using gradient backpropagation akin to the training procedure of neural networks. Gomez-Nieto et al. [GNCM*16] employ multidimensional projection and mixed integer optimization to preserve semantic relations and efficiently use display space in arrangements of geometric primitives. Small multiples feature grid views providing different perspectives on a single dataset [Tuf01, p. 170]. They can be used for interactive data exploration [vdEvW13] and have been applied to study flow maps [BBL12], movement and flow simulations [CFSL07], and biomechanical motion data [KERC09]. An extension by Meulemans et al. adds spaces when placing small multiples into a grid to improve visual clarity [MDS*17]. This is related to the concept of void members in this paper (see Sec. 3).

3. Level-of-Detail Grid Layout (LDG)

LDGs employ a hierarchical approach to scale distance-preserving grid layouts to large ensembles both visually and performance-wise. For effective level-of-detail representation, we target high similarity both of neighbors in the grid (enabling expressive exploration of similar members) and within branches in the hierarchy (yielding meaningful aggregates). In contrast, hierarchical clustering techniques (in combination with treemap presentations for example) solely focus on homogeneous groups without accounting for the spatial layout for the presentation (e.g., via treemaps, Fig. 2c), while traditional grid layouts only take neighborhood into account (see



(a) *node-link view of the LDG quadtree structure* **(b)** *grid at height* $h = 0$

Figure 3: *LDG for 63 members of the Turbo colormap and one grayscale color in (a) node-link view and (b) grid view at highest granularity. Nodes depict the (average) color of its member(s), node id (large, top), and normalized evaluation value* γ *(*$h = 0$*) or disparity* Δ *(*$h > 0$*).*

detailed discussion in Sec. 7). A previous hierarchical extension for Isomatch separately conducts layouting and member partitioning and exhibits significant shortcomings (e.g., Fig. 2b, Sec. 7). LDGs goal is not to partition the data into distinct clusters but yield one consistent grid representation suitable for hierarchical presentation.

Structure. LDGs employ a quadtree in which every node $n \in N$ has a corresponding quadratic area in the grid: nodes $N_h \subset N$ at height h cover $2^h \times 2^h$ cells (Fig. 3, $N_0 : 1, N_1 : 4, N_2 : 16$, etc.). Provided with the specified grid dimensions, a quadtree is created that is large enough to fully cover this grid. All leaf nodes $N_0 \subset N$ that are not part of the defined grid area are left unused (implemented by statically assigning so-called void members to them, see below). Throughout this paper, we enumerate heights and node ids starting from the leaves, e.g., for a grid of 64 cells, the node ids $\{0,\dots,63\}$ are at height $h = 0$, $\{64,\dots,79\}$ are on $h = 1$, $\{80,\dots,83\}$ are on $h = 2$, and finally 84 is the root node on $h = 3$ (Fig. 3a). Nodes are mapped to grid cells by recursively subdividing the grid area top-down from the root n_{root}, associating the top-left quadrant with the first child, the top-right quadrant with the second child, and the bottom-left and bottom-right quadrant with the third and fourth child, respectively (Fig. 3b). In Fig. 3, node 84 at $h = 3$ represents the leaf node range $\{0,1,\dots,63\}$, which is split for its children at $h = 2$ into the index ranges $\{0,\dots,15\}$ for node 80, $\{16,\dots,31\}$ for 81, $\{32,\dots,47\}$ at 82, and finally $\{48,\dots,63\}$ at 83. Here, node 80 represents the top-left quadrant, 81 the top-right quadrant, and 82 and 83 the bottom-left and bottom-right quadrant, respectively.

Optimization Objective. Assignment $A : N_0 \to T$ maps each leaf node $\in N_0$ to a member from data collection T. Inner nodes $n \in N_{>0}$ contain the aggregate $\bar{t}(n)$ of members at corresponding leaves $N_0(n)$. Below, $N_0(n) = \{\, n_0 \in N_0 \mid n \in N_\uparrow(n_0) \,\}$ denotes all leaves descending from n, whereas $N_\uparrow(n)$ provides all ancestors of n—nodes on the path to n_{root} (e.g., $N_0(72) = \{32,33,34,35\}$ and $N_\uparrow(32) = \{72,82,84\}$ in Fig. 3). LDG's objective function $\Gamma : A \to \mathbb{R}$ quantifies the associated cost of assignment A (lower is better), based on pairwise distances between members $d : T \times T \to \mathbb{R}$. Two goals are considered in optimizing A: **(G1)** homogeneous branches in LDG and **(G2)** similarity across neighboring nodes in the grid. To quantify homogeneity **(G1)**, LDG objective function Γ determines differences of members $t \in T$ to aggregates $\bar{t}(n)$ of inner nodes $n \in N_{>0}$ further up in the hierarchy. As a component of Γ, the evaluation value $\dot{\gamma} : N \times T \times A \to \mathbb{R}$ of assigning a member $t \in T$ to a node $n \in N_0$ is computed as follows under consideration of A:

$$\dot{\gamma}(n,t,A) = \sum_{n_\uparrow \in N_\uparrow(n)} d(t,\bar{t}(n_\uparrow)) \text{ with } \bar{t}(n_\uparrow) = \frac{\sum_{n \in N_0(n_\uparrow)} A(n)}{|N_0(n_\uparrow)|}. \quad (1)$$

Akin to other distance-based techniques, we further strive for high similarity across neighbors in the grid layout (**G2**). 4-neighborhood is considered throughout this paper, i.e., differences to the left, right, top and bottom neighbors (fewer at the grid boundary). Neighbors can be from different tree branches, e.g., neighbors of node 48 are nodes 26, 37, 49, and 50 in Fig. 3b. Cost function $\gamma : N \times T \times A \to \mathbb{R}$ incorporates **G2** as well as **G1** by not only assessing $\dot{\gamma}$ (Eq. 1) for a node n but for its neighbors $N_+(n)$ as well:

$$\gamma(n,t,A) = \dot{\gamma}(n,t,A) + \frac{1}{4} \sum_{n_+ \in N_+(n)} \dot{\gamma}(n_+,t,A). \quad (2)$$

The factor $1/4$ reflects that four neighboring branches are considered, and has been chosen to attribute equal weight to **G1** and **G2**. For efficiency, results obtained for different nodes during the traversal are cached, e.g., node 48 is evaluated against nodes 76, 83, and 84 only once. Finally, objective function $\Gamma : N_0 \times A \to \mathbb{R}$ simply sums up the evaluation values of all leaf nodes N_0 under assignment A:

$$\Gamma_{T,N_0}(A) = \sum_{n_0 \in N_0} \gamma(n_0, A(n_0), A). \quad (3)$$

Void members can be used in addition to regular ensemble members to fill grids and increase assignment flexibility. They are generally treated like regular members with two main differences: distance $d := 0$ when a void member is involved, and void members do not contribute to representations $\bar{t}$ at inner nodes (in Eq. 1, $|N_0(\cdot)|$ only counts leaves to which regular members are assigned).

Presentation. Each member is visually represented by a tile that covers the respective grid cell. Tiles of inner nodes (i) are sized to span a range of grid cells, and (ii) are further scaled to depict the ratio of regular to void members. Different aggregation schemes can be used to generate the representation at inner nodes, depending on the use case (see Sec. 5). Optimally, tiles expressively summarize comprised members, while also conveying inherent disparity. For example, the representation of channels in soil (Fig. 1) reflects commonalities of underlying members, but also directly conveys areas of high disparity. The stock use case (Fig. 2 (a–c)) depicts the range of individual trends. For some cases—like the Caltech image data base (Fig. 2 (d–e))—it may not be feasible to create such aggregates comprising all members $A(n) = \{A(n_0) \forall n_0 \in N_0(n)\}$ associated with a node n in a single tile, and instead the most central member $t \in T$ is chosen as representative: $\min_{t \in A(n)} \sum_{t^* \in A(n)} d(t,t^*)^2$.

Exploring the LDG essentially means changing at what granularity certain areas of the grid are visible (i.e., which nodes are shown). In this work, we consider rather basic interaction modalities, which

allow to adjust the set of visible nodes in three different ways. First, making all nodes of a specified height h visible yields good comparability across equally-sized tile images for an overview of member content—possibly reflecting disparity in tile representations (e.g., Fig. 1b). Second, directly interacting with a tile at height h allows to expand respective nodes to $h-1$ or collapse them to $h+1$ (i.e., substituting the tiles of four siblings by the tile of their parent, or vice versa). Third, disparity $\Delta(n)$ is a normalized measure of how well an aggregate representation $\bar{t}(n)$—the average in feature space of all members assigned to leaves of n (Sec. 4)—captures its members:

$$\Delta(n) = \frac{\Delta'(n)}{\Delta'(n_{\text{root}})} \text{ with } \Delta'(n) = \sum_{n_0 \in N_0(n)} d(A(n_0), \bar{t}(n)). \quad (4)$$

The measure is normalized via the disparity at n_{root}, the aggregate of all members. Visible nodes (whose tiles are shown in the grid presentation) are then determined by going from leaves to the root, collapsing nodes n—i.e., they become invisible and their parent node is set to visible—as long as their disparity $\Delta(n)$ does not exceed τ. Adaptive selection conveys an impression of the distribution of members: grid areas with more diverse members are shown in much higher detail than largely homogeneous areas.

4. LDG Optimization

LDGs are generated by finding an assignment A of tiles to grid cells (or leaf nodes) N_0 such that cost function Γ is minimized: $\min_A \Gamma(N_0, A)$. Conceptually, a specific assignment A is one element of the set of all possible permutations of tiles. In total, there are $|N_0|!/(|N_0|-|T|)!$ combinations, with $|N_0|$ being the number of cells in the grid, and $|T|$ denoting the amount of regular (non-void) tiles. Accordingly, it can be regarded as a combinatorial optimization problem, and even more concretely as an assignment problem. Linear assignment—which only considers the cost of assigning one element of one set to an element of another one—can be solved in polynomial time using the Hungarian algorithm [Kuh55]. In the quadratic assignment problem, the cost between assigned elements is considered as well, rendering the problem significantly more complex [KB55]—it has been proven to be NP-hard [SG76]. Our problem bears some resemblance to the quadratic assignment problem in that (virtually all) other assignments have an impact on the cost associated with the assignment of a single element due to the considered neighborhood and our hierarchical structure in particular. Such problems are typically infeasible to address with exact methods that find the optimal solution [MR01], and commonly heuristics are employed instead [AZ02]. Heuristics aim to identify a meaningful solution by exploring a promising part of the search space via so-called local search. Our local search approach for LDG generation independently considers subsets of the whole permutation and solves linear assignment problems within them.

Approach Outline. The problem of minimizing the objective function Γ is challenging as ensemble sizes can be in the order of millions and practically the full assignment A has an impact on the cost of a single member placement. Our progressive approach starts from randomly initialized A and repeatedly runs local search optimization passes to improve A. A pass starts with (randomly) selecting the hierarchy level h on which to exchange LDG nodes

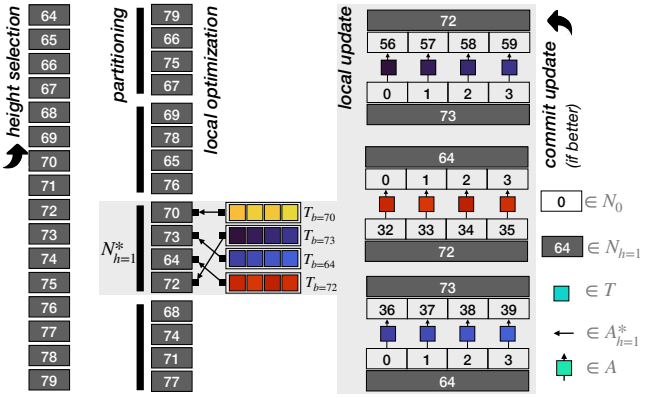


Figure 4: *Local search optimization: in each pass, select a level (here $h=1$), shuffle and partition respective nodes into sets $N^*_h \subset N_h$, optimize locally, and finally update the assignment (Alg. 1).*

N_h (*height selection*, Fig. 4). For local search, N_h is split into equally-sized partitions (*partitioning*), before optimizing the assignment within each considering the input assignment A (*local optimization*). A is not adapted during local optimization, but instead changes are reflected in a copy $\hat{A}$. $\hat{A}$ is adjusted by reassigning the 4^h members belonging to each node (*local update*), preserving their local order (essentially moving sub-grids). Finally, $\hat{A}$ replaces assignment A if it constitutes an improvement (*commit update* when $\Gamma(\hat{A}) < \Gamma(A)$).

Conceptually, a major "simplification" of this heuristic is that while optimizing the assignment in an iteration (to yield an updated version $\hat{A}$), it evaluates the cost γ for each node on the basis of the assignment A from the prior iteration. As we demonstrate throughout this paper, this approach yields high-quality, converging results. It has two major benefits that allow efficiently processing large data collections: (1) sub-problems in partitions can be solved independently in parallel to each other (avoiding conflicts by deferring updates to A to the end of a pass after local optimization), and (2) linear assignment can be used within each partition. Another crucial aspect is that the assignment is iteratively optimized at different heights h, utilizing the hierarchical structure of LDGs to operate on different levels of granularity: fine-granular changes (at low heights) can locally form smaller areas of similar members, while course-grained changes rearranges whole such areas to form larger homogeneous regions in the grid.

Below, the local search procedure is described in detail, before discussing parallel implementation, complexity, and termination.

Local Search. Each pass considers all LDG nodes N_h at height h, optimizing the assignment by exchanging 4^h members at once (Alg. 1). h is randomly chosen prior to each pass (*height selection* in Fig. 4), with a probability proportional to 2^{-h}, reflecting that larger numbers of nodes at lower heights require more adaptation (see Sec. 6). We generate a shuffled list of node indices $\tilde{N}_h$ for height h (Line 3) and loop over its partitions $N^*_h \subset N_h$ (Line 5, Line 6, *partitioning* in Fig. 4). The partition size k ultimately does not affect assignment quality but performance characteristics: it balances the computational cost of a pass—lower for smaller k—against the required number of passes–lower for larger k ($k=10$ is used in this work, Sec. 6). The assignment within each partition N^*_h is individually optimized. For this, we virtually set all leaves of involved

Algorithm 1 Local search pass for generating LDGs. A pass operates on height h, independently optimizing partitions $N_h^* \subset N_h$.

1: **function** LOCALSEARCHPASS(A, N, h, k)
2: ▷ consider all nodes N_h on height h in this optimization pass
3: $\tilde{N}_h \leftarrow \text{shuffle}(N_h)$
4: ▷ identify best assignments within partitions
5: **for all** $i \in \{0, k, 2k, 3k, \ldots, |\tilde{N}_h| - 1\}$ **do**
6: $N_h^* \leftarrow \{\tilde{N}_h[i], \ldots, \tilde{N}_h[i+k-1]\}$
7: ▷ (virtually) assign void members $\varnothing$ to involved leaves
8: $A^{\varnothing} \leftarrow \{A \mid A(n_0) = \varnothing \; \forall n_0 \in N_0(N_h^*)\}$
9: ▷ cost matrix C_l^* for assigning members $\in T_b$ to node n_a
10: **for all** $\{(n_a, n_b) \mid n_a, n_b \in N_h^*\}$ **do** ▷ loop over all node pairs
11: $T_b \leftarrow \{A(n_0) \mid n_0 \in N_0(n_b)\}$ ▷ members at leaf nodes
12: $C_l^*(n_a, T_b) \leftarrow \sum_{t_b \in T_b} \gamma(n_a, t_b, A^{\varnothing})$ ▷ evaluate $t_b \in T_b$ at n_a
13: ▷ solve linear assignment problem with cost matrix C_l^*
14: $A_l^* \leftarrow \text{linearAssignment}(C_l^*)$
15: ▷ update $\hat{A}$, maintain order among leaves of same node n_h
16: $\hat{A} \leftarrow \hat{A} \cup \{A(N_0(n_h)) \leftarrow A_h^*(n_h) \; \forall n_h \in N_h^*\}$
17: ▷ only use new $\hat{A}$ if it actually reduces cost w.r.t. A
18: **if** $\Gamma_{N_h,T}(\hat{A}) < \Gamma_{N_h,T}(A)$ **then** ▷ evaluate from height h upward
19: **return** $\hat{A}$
20: **else**
21: **return** A

nodes $N_0(N_h^*)$ to void members to remove their impact on node representations, yielding $A^{\varnothing}$ (Line 8). This avoids a bias toward the current state when considering changes to the assignment.

Matrix C_h^* depicts the cost of assigning the members T_b (currently assigned to node n_b, $|T_b| \leq 4^l$) to node n_a (Line 10–Line 12). With C_h^* a linear assignment problem is then solved to determine new member assignments (Line 14, *local optimization* in Fig. 4). Note that this addresses a simplified version of the actual problem even for an individual partition, as the influence of the other assignments within that partition is neglected. With the cubic complexity of linear assignment, this is significantly cheaper and eventually more efficient than checking all possible permutations to account for mutual influence. The changes determined for inner nodes are transferred down to leaves for new assignment $\hat{A}$, maintaining the local order of members (i.e., exchanging whole sub-grids, Line 16, *local update* in Fig. 4). When the assignment is well-refined already in later stages, it can occur that $\hat{A}$ is actually worse than A due to the employed simplifications. To quantify this, we compare the respective Γ from the chosen height h upward and keep the assignment with the lower score (Line 18–Line 21, *commit update* in Fig. 4).

Implementation and Complexity. The local search procedure can be interrupted after each pass, e.g., after a time budget is exhausted or when the rate of improvement has become slow, indicating convergence (see Sec. 6). In practice, the application context and its constraints are essentially the decisive factors to choose a suitable criterion. Partitions N_h^* can be processed in parallel without incurring read-write conflicts: different subsets $N^* \subset N_h$ do not overlap, and changes during a pass are applied to a copy $\hat{A}$ of the assignment and not the input assignment A which is considered for the evaluation within partitions (Alg. 1, Line 8 & Line 11). This paper uses multicore CPUs to be less constrained in terms of memory, but large ensembles would otherwise be a great fit for GPUs (implementation and evaluation are planned for future work). We use a 32-bit Mersenne Twister [MN98]—an equidistributed uniform pseudo-random number generator—for generating initial assignments, shuffling nodes (Alg. 1, Line 3) and selecting heights h prior to each local pass.

The computational complexity of a pass on height h is

$$O\left(|N_h|\,(k \log_4(|N_h|)\,|F_T|\,\frac{|T|}{|N_0|} + k^2)\right), \tag{5}$$

i.e., it scales quasilinearly with the number of nodes $|N_h|$, quadratically with partition size k, and linearly with the size of a member's feature representation $|F_T|$. It contains the number of partitions (in $O(N_h/k)$), the complexity of the Hungarian Algorithm ($\approx O(k^3)$), and the cost for evaluation γ ($O(|F_T| \log_4|N_0|)$. An approximately linear impact is practically measured for the comparably small values of $k(\in \{2, \ldots, 40\})$ considered in this work (Sec. 6), i.e., the corresponding linear term in Eq. 5 appears to be more influential in this scenario. The cost of distance function $d(\cdot,\cdot)$ is in $O(|F_T|)$ for both the Euclidean and the Cosine distance employed in this work; the ratio $|T|/|N_0|$ of ensemble size to grid size reflects that no distance computations are issued for void members). Space complexity is determined by the required storage for aggregate representations of all nodes N: $O(|F_T|\,|N|)$.

5. Use Cases

LDGs are now applied to three diverse use cases: an image dataset with extra feature representation from deep learning [LFP04, LFP06] (Sec. 5.1), a million stock price predictions from a mathematical finance model (Sec. 5.2), and channel structures in soil from Markov chain Monte Carlo (MCMC) with domain scientist feedback (Sec. 5.3). A simple interactive web viewer—ported from a Qt desktop application—allows readers to practically explore LDGs: `https://ldg-demo.github.io`.

5.1. Image Collection with ML-based Feature Space

We now apply LDGs to image collection Caltech 101 [LFP04, LFP06]). The collection contains 9144 images ($\approx 300 \times 200$ px), featuring 101 attributed categories like faces, airplanes, watches, pianos, etc. A color frame indicating its labeled category is added for the largest 19 groups (others have a gray frame, categories are underlined accordingly in the text). Images within a category can be highly diverse—in particular the background category contains a wide variety including foods, landscapes, boardgames, animals, illustrations, icons, etc. — or show significant overlap in content (like faces and faces easy). Crucially, this categorical information is not used directly in our optimizer. Instead, the values at the layer prior to the output of a Convolutional Neural Network (CNN) classifier trained with these labels serves as expressive feature representation F_T ($|F_T| = 2048$) of the image's content for LDG generation. Similarity is assessed via the cosine distance measure: $d_{\cos} = 1 - \frac{f_a \cdot f_b}{|f_a||f_b|}$ ($f_a, f_b \in F_T$). Members are placed in a grid layout of 80×128 (9144 fixed and $10240 - 9144 = 1096$ flexibly placed void members). The image whose feature vector has the smallest total distance to all others serves as tile representative at inner nodes (akin to Fried et al. [FDH*15]).

(a) *height* $h = 0$ **(b)** *height* $h = 3$ **(c)** *adaptive selection* ($\tau = 0.5$) **(d)** *zoom & refine from (c) to* $h = 1$

Figure 5: *LDG for Caltech 101. (a) Grids with* 9144 *visible nodes on* $h = 0$ *exhibit too many and too small tiles for an expressive summary. (b)* $h = 3$ ($4^3 = 64$ *members/node) yields more clarity, yet diverse images may be summarized in one representation. (c) Adaptive selection collapses homogeneous areas and expands diverse regions. (d) Investigation of big cats down to* $h = 1$ *from (c) (black rectangle, center right).*

Images from the same category are typically positioned to be close in the grid and the quadtree hierarchy, e.g., airplanes or motorbikes (Fig. 5). This can also apply to similar images from different categories, like faces and faces easy, or helicopters and airplanes, whereas images from the diverse background category are scattered and positioned in the vicinity of similar neighbors of other categories throughout the grid. This demonstrates the emergence of similar features for similar objects during training, and indicates the benefit of using feature vectors to organize members beyond their original categorization, eventually allowing LDGs to expressively reflect image similarity. For instance, zooming into Fig. 5a demonstrates this for mammals (e.g., wild cats, cougar, kangaroo, dalmatian, etc.), sea animals (e.g., dolphins, bass) or flowers (sunflower, lotus, water lily). Watches and inline skates are placed next to motorbikes and wheelchairs, presumably due to predominant circular shapes. Void members are generally positioned between groups of similar images.

While height $h = 0$ provides a view that is too fine-granular for large collections (Fig. 5a), $h = 3$ gives a much clearer quick overview on the dataset (Fig. 5b). Adaptive node selection reflects the similarity of assigned members and further expands visible nodes in more heterogeneous grid areas (see Fig. 5c). Homogeneous areas are represented by large tiles, whereas more heterogeneous regions exhibit higher granularity. It can clearly be seen that there are around a thousand airplanes, faces, and motorbikes (the largest tiles belonging to visible nodes are on $h = 4$, i.e., representing up to 256 images). This also serves as a meaningful basis for further exploration, as a quick overview can be gained on predominant types, while also reflecting more diverse grid areas. Fig. 5d explores the ≈ 300 big cats from Fig. 5c more closely by expanding respective visible nodes to $h = 1$. Among others, it can be seen that leopards—in addition to being close to related categories like cougars, and wildcats—are also located next to crocodiles. The shared occurrence of spotted patterns and the pictures of leopards at waterholes are possible explanations for their apparent similarity in feature vectors.

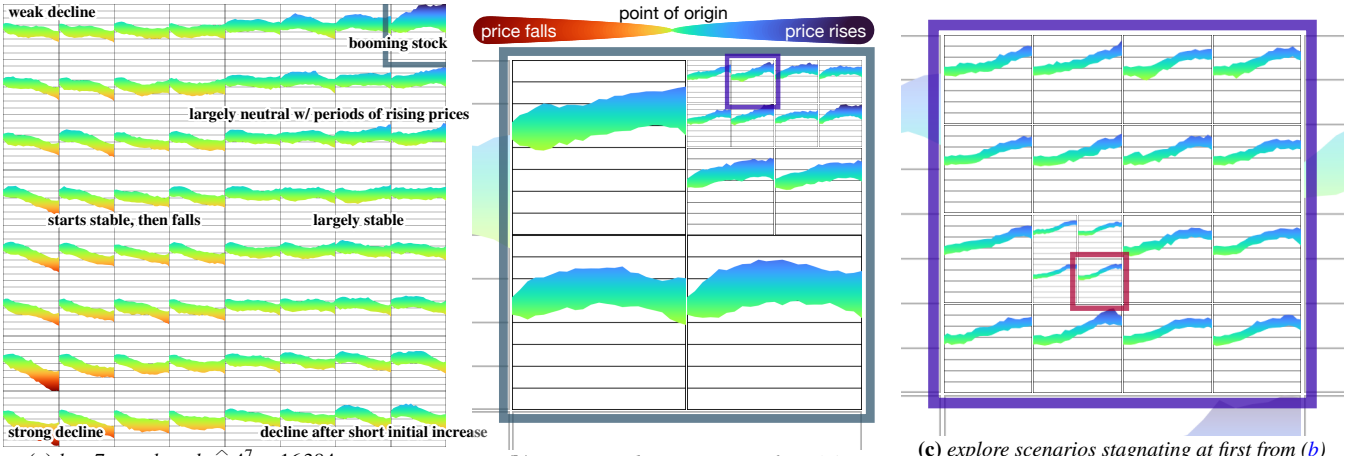
5.2. Stock Price Simulation

Geometric Brownian motion is commonly used in mathematical finance to predict stock prices via the Black–Scholes model [BS73]—considering long-term trends and stochastic shorter-term fluctuations. A large prediction ensemble allows model-based assessment of different outcomes and their likelihood. This use case considers 1 048 576 (4^{10}) predictions for German electric utility company E.ON for the 23 working days in August 2019 (Fig. 6), based on Xetra Exchange data from the previous month (Fig. 7). The LDG has a total height of 10, no void tiles are used. Stock prices are compared via Euclidean distance, tiles depict the range of underlying trends.

Fig. 6a provides a comprehensive overview on stock predictions with visible nodes at height $h = 7$, i.e., conveying $4^7 = 16\,384$ outcomes each. There is a variety of characteristically different stock developments (cf. annotations in Fig. 6a), with the majority exhibiting a slight downward trend, essentially reflecting the development observed in July. However, there is also a fraction of cases that depict a soaring stock with significant gains (rectangle in Fig. 6a), which is expanded for a closer look in Fig. 6b. The share of cases that at the end of August approach the stock's maximum price is considered in particular. Interestingly, when drilling down further, tiles indicate that some price developments with significant gains at the end of the month even exhibit a (mild) downward trend in the first couple of days. Exploring this down the LDG to the level of individual model predictions in Fig. 6c shows that there are indeed a couple of developments starting slow but eventually reaching high prices. One member (highlighted in Fig. 6c) even closely approaches the maximum price of the whole ensemble toward the end of August despite stagnating in the first couple of days.

5.3. Channel Structures in Soil

This use case considers an ensemble of 95 000 channel structures in soil that has been obtained via Markov chain Monte Carlo (MCMC).



(a) $h = 7$ *: each node* $\widehat{=} 4^7 = 16\,384$ *outcomes* **(b)** *investigate booming prices from (a)* **(c)** *explore scenarios stagnating at first from (b)*

Figure 6: *One million stock predictions. (a) From an overview on* $h = 7$ *, (b,c) rising stock prices are explored to individual predictions* ($h = 0$).

Figure 7: *Original stock prices obtained from Xetra Exchange for a German electric utility company (E.ON) from July 2019 (note that the depicted stock price range differs from charts in Fig. 6).*

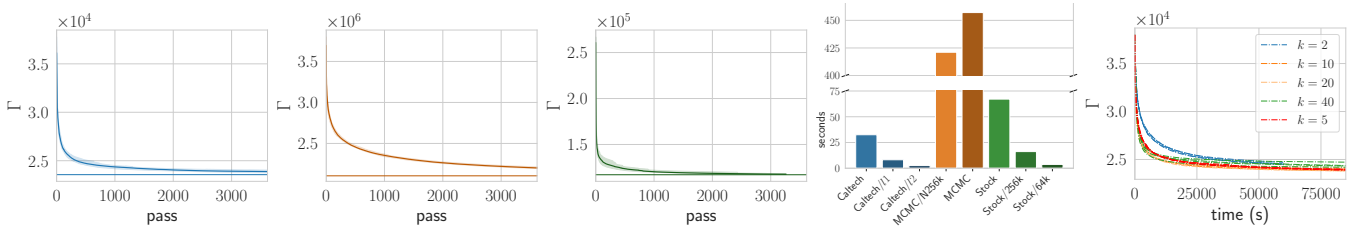
MCMC conducts inverse calculations of channel structures based on sparse, non-transient hydraulic head measurements. Each sample consists of a grid of 50×50 scalar values that represent the estimated hydraulic conductivity of the soil. Members are placed in a LDG of height 9 and a 256×512 grid layout (i.e., there are 131 072 fixed and $131\,072 - 95\,000 = 36\,072$ flexibly placed void members). We collected feedback from a domain scientist who generated and analyzed this data in his research. Currently, the standard evaluation approach is to initially study (i) the variation in channel geometry for a small selection of members and (ii) the average across all members to assess the probabilities of channels at different locations (our aggregation scheme for tiles reflects this accordingly, Fig. 1). The expert further employs k-means clustering, for which two main issues were noted: (1) choosing an adequate number of clusters, and (2) significantly different clustering results for similar data.

The expert notes that LDGs can improve the analysis in different ways. LDGs can handle large ensembles visually as well as performance-wise, and yield stable results for a reliable analysis (Sec. 6). They provide a comprehensive summary of structures from MCMC, and selecting visible nodes at different heights or via adaptive selection is useful for further exploration (e.g., Fig. 1). Highly aggregated views showing fewer tiles are generally preferred by our expert: he notes that viewing too many tiles at once induces a high cognitive load and is less effective for his analysis.

6. Performance Analysis

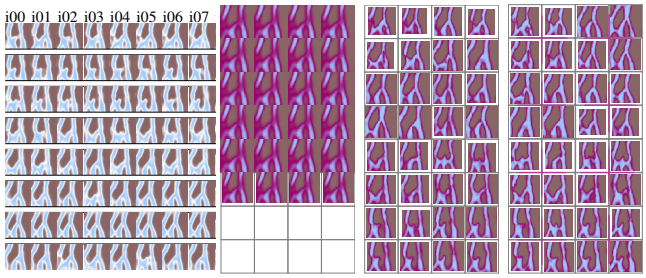
LDG's performance is now investigated via 16 individual runs for each case conducted using an AMD Ryzen 7 2700X 8-Core Processor and 32 GB RAM. Fig. 8a–c shows that progressive refinement curves qualitatively depict similar development toward convergence: they start with rapid significant improvements, significantly slowing down closer to the optimum. The comparably narrow range around the median in the plots indicates the good-natured refinement behavior that yields qualitatively comparable results regardless of the seed initializing the pseudo-random number generator. Results from the eight runs with the lowest seeds from Fig. 8b are compared in Fig. 9a (tiles from one run are organized in one column i00–07): while there naturally is some variation, the basic structures are similar and demonstrate no significant impact of stochastic factors in the result. This is also reflected in the respective Γ-values. Results at different stages of refinement for one run are shown in Fig. 9b–d. The initial random assignment yields similar aggregated results across nodes with high disparity (b, with cells in bottom rows only featuring void members). 32 passes already significantly reduce disparity and clear structures emerge in tiles (c), yielding a structure that remains largely stable (at $h = 6$) as predominantly fine-granular, local adaptations occur (d). The LDGs presented in Sec. 5 are the results with the lowest Γ-values from this study with 12352 refinement passes for Caltech, 4224 for MCMC, and 5696 for Stock. This almost directly reflects the number of conducted passes per case in the experiments: while improvements are insignificant at later stages as discussed above, Γ still slightly decreases at slow rates.

As discussed in Sec. 4, the number of members—see Stock study in Fig. 8d—as well as the number of elements in a member's feature representation—compare across use cases in Fig. 8d—have a practically linear impact on the cost of an individual pass. Furthermore, each increase in height h deceases the number of considered nodes by $\approx \times 4$, which is also linearly reflected in lower costs—see Caltech in (d). The variant MCMC/N256k with more void members but the same quadtree height compared to the standard MCMC configuration yields faster LDG computation as the higher sparsity of the tree reduces the number of distance computations. The very narrow (barely visible) black error bars also indicate stable pass costs. Fig. 8e shows by example that the range of partition sizes $k = 5, 10, 20$ yields similarly good performance, with smaller $k = 2$ and larger $k = 40$ performing worse. We use $k = 10$ throughout this work. k appears both as linear and quadratic term in Eq. 5. Practically, we observe a roughly linear growth of average pass costs for the comparably small partition sizes considered in this work ($k = 2 : 12.4\,\text{s}, k = 5 : 18.6\,\text{s}, k = 10, 28.7\,\text{s}, k = 20 : 52.2\,\text{s}, k = 40 : 108.4\,\text{s}$).



(a) *Caltech* **(b)** *MCMC* **(c)** *Stock* **(d)** *local search pass costs* **(e)** *partition size k*

Figure 8: *16 LDG generation runs each for use cases and variants: Stock/64k and Stock/256k feature $|T| = 64k$ and 256k tiles, respectively (original: $|T| = 1024k$); MCMC/N256k uses $|N_0| = 256k$ (instead of $|N_0| = 128k$). (a–c) Refinement via local search passes (range is depicted by filled area, thick line provides the median, thin horizontal line indicates best known solution after extensive runs), and (d) local search pass timings, depicting local search pass costs on different levels for Caltech (black error bars show variation). (e) Impact of partition size k.*



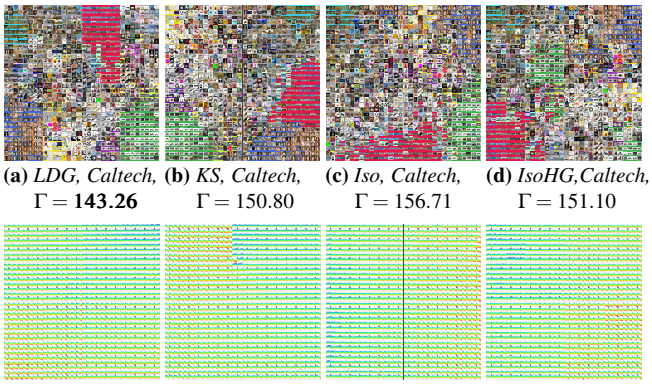
(a) *a column per run* **(b)** ***0*** *(3.81e6)* **(c)** ***32*** *(2.87e6)* **(d)** ***3520****(2.2e6)*

Figure 9: *Impact of stochastic factors. (a) Aggregate representations of eight individual runs i00–i07 at height $h = 7$. Results are similar throughout despite using different seeds (also reflected in similar values for Γ, see Fig. 8b). (b–d) Refinement results at $h = 6$ for an MCMC run, demonstrating quick convergence on higher levels. A modified color map emphasizes ambiguity (pink replaces white).*

While taking longer with increasing k, this also allows the optimizer to achieve larger improvements on average ($k = 2 : 4.6\Delta\Gamma, k = 5 : 16.7\Delta\Gamma, k = 10, 33.3\Delta\Gamma, k = 20 : 61.3\Delta\Gamma, k = 40 : 88.9\Delta\Gamma$). For larger k, the timings indicate a trend toward superlinear growth, while the achieved average improvement $\Delta\Gamma$ increases more slowly.

7. Comparison to Other Grid Layout Approaches

The major conceptual difference of LDGs to previous techniques is the consideration of hierarchy during placement (**G1**, Sec. 3) in addition to neighborhood (**G2**). Accordingly, LDGs yield a novel—yet related—grid representation, with Γ serving as corresponding measure for quantitative assessment (lower ≙ better). All layouts—across LDG, Kernelized Sorting [QSST10], Isomatch [FDH*15], as well as extension IsoMatchHG (see description below)—generally exhibit high similarity of neighboring members (Fig. 10). In all Caltech results, airplanes, motorbikes, as well as faces are located next to each other (Fig. 10b–c). This also holds true for the Stock predictions, placing tiles depicting similar trends in close proximity (Fig. 10f–g). LDGs occasionally even yield locally smoother results than alternatives, e.g., there is no sharp transition in Fig. 10e like that of Fig. 10f in the top center between rising and falling trends. IsoMatch expands rising trends along the left border, resulting in similar members being located comparably far away (Fig. 10g). Besides neighborhood, the additional consideration of branch homogeneity can also clearly be seen for LDG (Fig. 10a and e).



(a) *LDG, Caltech,* $\Gamma =$ **143.26** **(b)** *KS, Caltech,* $\Gamma = 150.80$ **(c)** *Iso, Caltech,* $\Gamma = 156.71$ **(d)** *IsoHG,Caltech,* $\Gamma = 151.10$

(e) *LDG, Stock,* $\Gamma =$ **328.44** **(f)** *KS, Stock,* $\Gamma = 363.40$ **(g)** *Iso, Stock,* $\Gamma = 369.76$ **(h)** *IsoHG, Stock,* $\Gamma = 345.11$

Figure 10: *Comparison of LDG against Kernelized Sorting (KS) [QSST10] and IsoMatch (Iso) [FDH*15] for downsampled Caltech and Stock (with $|T| = 1024, h = 0$). LDG (a & e) exhibits neighborhood similarity comparable to previous approaches (b, c, f, g), but additionally considers hierarchies. A subset of $|T| = 1024$ members is used as both KS and Iso exhibit approximately cubic complexity regarding $|T|$, resulting in issues for larger $|T|$.*

For both Kernelized Sorting and IsoMatch extensions have been proposed for better visual scaling with collection sizes exceeding several hundreds. Kernelized Sorting simply replaces the 2D grid with a 3D pyramidal version, also assigning individual members to leaves (crucially yielding no hierarchical organization with inner nodes representing their children) [QSST10]. For IsoMatch, Fried et al. [FDH*15] proposed an extension for hierarchical visualization. Starting from a node n_{h+1} (initially the root node containing all members T), their top-down method iteratively creates m child nodes $(n_h^{c[0]}, n_l^{c[1]}, \ldots, n_h^{c[m-1]})$ by splitting respective members into equally-sized clusters. Then, the grid layout for node n_{h+1} is generated by assigning the m members of $n_l^{c[0\ldots m-1]}$ closest to cluster centroids to grid cells (also using respective images as representatives for each child). The grid layout is created individually for each node, independent from the layouts of parents, children or siblings. This means that only a single node n_{h+1} at one fixed aggregation height h can be shown at a time, as grid layouts of any pair of nodes are not compatible (regardless of whether they are siblings at the same height or one is a child of the other). LDGs, in contrast, provide visual summaries of the full data with visible nodes at arbitrary

heights in one consistent view. To the best of our knowledge, no other grid layout technique achieves this. A variant of the hierarchical extension of Fried et al. is now introduced for a meaningful comparison: IsoMatchHG. It generates the lower-dimensional embedding for grid layouting only once for the full data, and re-uses it at every node instead of creating a new one like IsoMatch. This yields significantly more consistent placement, but does not affect clustering itself (i.e., the assignment of members to nodes remains unchanged to the original). We use $m = 4$ for direct comparability with the quadtree structure of LDGs. Fig. 10d & h show that IsoMatchHG clearly reflects a hierarchical grid structure akin to our approach—e.g., assigning airplanes and faces to the $h = 3$-node in (d)—which is also indicated by the lower Γ-score in comparison to IsoMatch. Still, across both Caltech and Stock, the score associated with LDGs is significantly lower than any alternative including IsoMatchHG. Artifacts in IsoMatchHG can clearly be seen as well: when the clustering splits similar members early on higher up in the hierarchy as it needs to produce equally-sized groups, they are also positioned far apart in the grid (e.g., motorbikes). Fig. 2 presents grids for LDG and IsoMatchHG at different granularity, and exemplifies that hierarchical clustering in combination with treemaps (HC+TM) for visualization also results in high discontinuities in the layout as well as separated groups of similar members (besides being less efficient in terms of visual space). Conceptually, this can be attributed to the fact that HC+TM considers the clustering of members for generating the hierarchy separately from generating the layout. Critically, the difference to our LDG approach is that we do not aim to cluster the data—which might be considered an ill-posed task for ensembles with smoothly varying members. Instead, LDGs use a hierarchical grid structure for level-of-detail presentation, which is generated by jointly optimizing for high member similarity within hierarchically grouped cells and across grid neighbors.

8. Discussion and Conclusion

LDGs provide summaries of large data collections with millions of members and can be generated efficiently solely based on mutual distances. The level-of-detail grid representation yields visual scalability, and the local search-based parallel progressive optimization scheme explicitly exploits the hierarchical structure for computational scalability. The hierarchical design enables interactive investigation via adaptive selection or manual node expansion and always yields a complete, consistent grid view presenting all members at different levels of granularity. To the best of our knowledge, LDG's ability to generate and present large consistent grid layouts exceeds that of prior grid layout-based techniques by orders of magnitude. LDG's utility has been exemplified via diverse use cases: image collections with feature vectors from a neural network, a million stock market predictions from a Black-Scholes model, and 95 000 channel structures in soil from Markov chain Monte Carlo.

The major focus of this work is on the optimization approach to efficiently generate LDGs for large data collections. This is achieved by the adoption of a local search strategy in combination with a custom-tailored hierarchical scheme that allows to optimize the grid at different granularity levels h. We were able to empirically demonstrate favorable characteristics in our evaluation. However, no formal guarantees regarding convergence characteristics can be provided, besides that after each local search pass the result is better or at least equal in terms of evaluation score Γ (Alg. 1 Line 18–Line 21). LDGs yield similar refinement behavior (Fig. 8a–c) and resulting grids (Fig. 9a) for different random initialization. It would be an interesting direction for future work to investigate whether other initialization schemes might yield good results more quickly and whether they introduce biases of some sort. Our approach further follows a progressive approach in the sense that it iteratively refines results, and we could additionally cater to streaming scenarios with data incoming at runtime by placing new elements into an extended grid space (or incorporating them within the current grid if enough void space is available). Visual analysis approaches could generally be useful to understand how such extensions impact refinement characteristics [WKF21]. An important strength of LDGs is that they can be flexibly explored, and basic means for interaction are considered in this work: via global settings regarding height h or disparity Δ, as well as directly by selecting tiles for expansion or collapse. We believe that dedicated future work on interaction concepts for LDGs has great potential to improve the exploration process. For instance, disparity could be used as a basis for adaptive interaction beyond the specification of a global threshold τ, e.g., expanding tiles in the neighborhood of a selected tile dependent on whether they represent similar members. Tile representations already reflect the similarity across neighbors, but still additional visual cues could be beneficial to further emphasize this aspect. For this, we will look into dedicated disparity markers—e.g., lines between tiles with varying length or thickness—as well as markerless approaches (like shifting tiles toward similar neighbors). In this paper, LDGs have been demonstrated to provide comprehensive visual summaries, but they might also be useful for other tasks, like search for specific members in large ensembles, comparison of member pairs, or identifying outliers. This will be further investigated and potentially dedicated extensions added in future work. LDGs could further be embedded into a full-fledged (progressive) visual analytics framework [SPG14], integrating it with other visualization and analysis methods via brushing and linking.

Another objective for future work is to further evaluate and improve the computational scaling of LDG's placement optimization scheme toward even larger ensembles with billions of members. For this, an out-of-core approach will be required to circumvent memory limitations. This would also alleviate the major drawback of an additionally planned GPU implementation, which would otherwise be well-suited to exploit the massive degree of parallelism exhibited by the optimizer, presumably significantly accelerating LDG generation. The efficiency of the method itself could be improved as well, e.g., via heuristics for clever partitioning prior to local search. Our local search-based heuristic could further be combined with advanced population-based metaheuristics considering multiple candidate solutions, like ant colony optimization, evolutionary computation, particle swarm optimization, genetic algorithms, etc. [BR03] (such approaches have previously been applied successfully to other types of permutation-based problems [Meh11]).

Acknowledgments

Supported by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project Number 327154368 – SFB 1313.

References

[AZ02] ANGEL E., ZISSIMOPOULOS V.: On the hardness of the quadratic assignment problem with metaheuristics. *Journal of Heuristics 8*, 4 (July 2002), 399–414. doi:10.1023/A:1015454612213. 5

[BBL12] BOYANDIN I., BERTINI E., LALANNE D.: A qualitative study on the exploration of temporal changes in flow maps with animation and small-multiples. *Computer Graphics Forum 31*, 3pt2 (2012), 1005–1014. doi:10.1111/j.1467-8659.2012.03093.x. 3

[BHM16] BARTHEL K. U., HEZEL N., MACKOWIAK R.: Navigating a graph of scenes for exploring large video collections. In *MultiMedia Modeling*. Springer International Publishing, 2016, pp. 418–423. doi:10.1007/978-3-319-27674-8_43. 3

[BR03] BLUM C., ROLI A.: Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys 35*, 3 (Sept. 2003), 268–308. doi:10.1145/937503.937505. 10

[BS73] BLACK F., SCHOLES M.: The pricing of options and corporate liabilities. *Journal of Political Economy 81*, 3 (1973), 637–654. doi:10.1086/260062. 2, 7

[BSW98] BISHOP C. M., SVENSÉN M., WILLIAMS C. K. I.: GTM: The Generative Topographic Mapping. *Neural Computation 10*, 1 (01 1998), 215–234. doi:10.1162/089976698300017953. 3

[CFSL07] CHEN J., FORSBERG A. S., SWARTZ S. M., LAIDLAW D. H.: Interactive multiple scale small multiples. In *Poster Compendium of IEEE VIS 2007* (2007), pp. 46–47. 3

[CLG16] CAO N., LIN Y., GOTZ D.: Untangle map: Visual analysis of probabilistic multi-label data. *IEEE Transactions on Visualization and Computer Graphics 22*, 2 (Feb 2016), 1149–1163. doi:10.1109/TVCG.2015.2424878. 3

[DSF*14] DUARTE F. S. L. G., SIKANSI F., FATORE F. M., FADEL S. G., PAULOVICH F. V.: Nmap: A novel neighborhood preservation space-filling algorithm. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (Dec 2014), 2063–2071. doi:10.1109/TVCG.2014.2346276. 3

[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. 2, 3, 6, 9

[GNCM*16] GOMEZ-NIETO E., CASACA W., MOTTA D., HARTMANN I., TAUBIN G., NONATO L. G.: Dealing with Multiple Requirements in Geometric Arrangements. *IEEE Transactions on Visualization and Computer Graphics 22*, 3 (Mar. 2016), 1223–1235. doi:10.1109/TVCG.2015.2489660. 3

[GRP*14] GOMEZ-NIETO E., ROMAN F. S., PAGLIOSA P., CASACA W., HELOU E. S., DE OLIVEIRA M. C. F., NONATO L. G.: Similarity preserving snippet-based visualization of web search results. *IEEE Transactions on Visualization and Computer Graphics 20*, 3 (March 2014), 457–470. doi:10.1109/TVCG.2013.242. 3

[HP19] HILASACA G. M. H., PAULOVICH F. V.: Distance preserving grid layouts. *CoRR abs/1903.06262* (2019). arXiv:1903.06262. 3

[JCC*11] JOIA P., COIMBRA D., CUMINATO J. A., PAULOVICH F. V., NONATO L. G.: Local affine multidimensional projection. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (Dec 2011), 2563–2571. doi:10.1109/TVCG.2011.220. 3

[Joh67] JOHNSON S. C.: Hierarchical clustering schemes. *Psychometrika 32*, 3 (Sept. 1967), 241–254. doi:10.1007/bf02289588. 3

[KB55] KOOPMANS T. C., BECKMANN M. J.: *Assignment Problems and the Location of Economic Activities*. Cowles Foundation Discussion Papers 4, Cowles Foundation for Research in Economics, Yale University, 1955. URL: https://ideas.repec.org/p/cwl/cwldpp/4.html. 5

[KERC09] KEEFE D., EWERT M., RIBARSKY W., CHANG R.: Interactive coordinated multiple-view visualization of biomechanical motion data. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (Nov 2009), 1383–1390. doi:10.1109/TVCG.2009.152. 3

[Koh90] KOHONEN T.: The self-organizing map. *Proceedings of the IEEE 78*, 9 (1990), 1464–1480. doi:10.1109/5.58325. 3

[Kuh55] KUHN H. W.: The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly 2*, 1-2 (1955), 83–97. doi:10.1002/nav.3800020109. 3, 5

[LFP04] LI FEI-FEI, FERGUS R., PERONA P.: Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories. In *2004 Conference on Computer Vision and Pattern Recognition Workshop* (June 2004), pp. 178–178. doi:10.1109/CVPR.2004.383. 6

[LFP06] LI FEI-FEI, FERGUS R., PERONA P.: One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence 28*, 4 (April 2006), 594–611. doi:10.1109/TPAMI.2006.79. 6

[LHNS18] LIU X., HU Y., NORTH S., SHEN H.-W.: CorrelatedMultiples: Spatially coherent small multiples with constrained multi-dimensional scaling. *Computer Graphics Forum 37*, 1 (2018), 7–18. doi:10.1111/cgf.12526. 3

[MDS*17] MEULEMANS W., DYKES J., SLINGSBY A., TURKAY C., WOOD J.: Small multiples with gaps. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (2017), 381–390. doi:10.1109/TVCG.2016.2598542. 3

[Meh11] MEHDI M.: *Parallel Hybrid Optimization Methods for Permutation Based Problems*. Theses, Université des Sciences et Technologie de Lille - Lille I, Oct. 2011. 10

[MHM18] MCINNES L., HEALY J., MELVILLE J.: UMAP: Uniform manifold approximation and projection for dimension reduction, 2018. arXiv:1802.03426. 3

[MN98] MATSUMOTO M., NISHIMURA T.: Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation 8*, 1 (Jan. 1998), 3–30. doi:10.1145/272991.272995. 6

[MR01] MANTACI R., RAKOTONDRAJAO F.: A permutations representation that knows what "Eulerian" means. *Discrete Mathematics and Theoretical Computer Science 4*, 2 (2001), 101–108. 5

[POdAL10] PINHO R. D., OLIVEIRA M. C. F., DE ANDRADE LOPES A.: An incremental space to visualize dynamic data sets. *Multimedia Tools and Applications 50* (2010), 533–562. 3

[PTD*21] PAN X., TANG F., DONG W., MA C., MENG Y., HUANG F., LEE T.-Y., XU C.: Content-Based Visual Summarization for Image Collections. *IEEE Transactions on Visualization and Computer Graphics 27*, 4 (Apr. 2021), 2298–2312. doi:10.1109/TVCG.2019.2948611. 3

[QKTB10] QUADRIANTO N., KERSTING K., TUYTELAARS T., BUNTINE W. L.: Beyond 2d-grids: A dependence maximization view on image browsing. In *Proceedings of the International Conference on Multimedia Information Retrieval* (New York, NY, USA, 2010), MIR '10, Association for Computing Machinery, p. 339–348. doi:10.1145/1743384.1743440. 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. 2, 3, 9

[RXN20] REUSCHEN S., XU T., NOWAK W.: Bayesian inversion of hierarchical geostatistical models using a parallel-tempering sequential Gibbs MCMC. *Advances in Water Resources 141* (July 2020), 103614. doi:10.1016/j.advwatres.2020.103614. 2

[SDW09] SLINGSBY A., DYKES J., WOOD J.: Configuring Hierarchical Layouts to Address Research Questions. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (Nov. 2009), 977–984. doi:10.1109/TVCG.2009.128. 3

[SG76] SAHNI S., GONZALEZ T.: P-complete approximation problems. *J. ACM 23*, 3 (1976), 555–565. doi:10.1145/321958.321975. 5

[SG14] STRONG G., GONG M.: Self-sorting map: An efficient algorithm for presenting multimedia data in structured layouts. *IEEE Transactions on Multimedia 16*, 4 (June 2014), 1045–1058. doi:10.1109/TMM.2014.2306183. 2, 3

[SHS11] SCHULZ H., HADLAK S., SCHUMANN H.: The Design Space of Implicit Hierarchy Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics 17*, 4 (Apr. 2011), 393–411. doi:10.1109/TVCG.2010.79. 3

[SMS*20] SONDAG M., MEULEMANS W., SCHULZ C., VERBEEK K., WEISKOPF D., SPECKMANN B.: Uncertainty Treemaps. In *2020 IEEE Pacific Visualization Symposium (PacificVis)* (June 2020), pp. 111–120. doi:10.1109/PacificVis48177.2020.7614. 3

[SPG14] STOLPER C. D., PERER A., GOTZ D.: Progressive visual analytics: User-driven visual exploration of in-progress analytics. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (2014), 1653–1662. doi:10.1109/TVCG.2014.2346574. 10

[SSS*12] STROBELT H., SPICKER M., STOFFEL A., KEIM D., DEUSSEN O.: Rolled-out wordles: A heuristic method for overlap removal of 2d data representatives. *Computer Graphics Forum 31* (2012), 1135–1144. doi:10.1111/j.1467-8659.2012.03106.x. 3

[TSL00] TENENBAUM J. B., SILVA V. D., LANGFORD J. C.: A global geometric framework for nonlinear dimensionality reduction. *Science 290*, 5500 (2000), 2319–2323. doi:10.1126/science.290.5500.2319. 3

[Tuf01] TUFTE E. R.: *The Visual Display of Quantitative Information*, 2 ed. Graphics Press, Cheshire, CT, 2001. 3

[vdEvW13] VAN DEN ELZEN S., VAN WIJK J. J.: Small multiples, large singles: A new approach for visual data exploration. *Computer Graphics Forum 32* (2013), 191–200. doi:10.1111/cgf.12106. 3

[vH08] VAN DER MAATEN L., HINTON G.: Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research 9* (2008), 2579–2605. 3

[WD08] WOOD J., DYKES J.: Spatially Ordered Treemaps. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (Nov. 2008), 1348–1355. doi:10.1109/TVCG.2008.165. 3

[WHLS19] WANG J., HAZARIKA S., LI C., SHEN H.: Visualization and visual analysis of ensemble data: A survey. *IEEE Transactions on Visualization and Computer Graphics 25*, 9 (Sep. 2019), 2853–2872. doi:10.1109/TVCG.2018.2853721. 1

[WKF21] WATERINK E., KOSINKA J., FREY S.: Visual Analysis of Popping in Progressive Visualization. In *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference* (2021), Frosini P., Giorgi D., Melzi S., Rodolà E., (Eds.), The Eurographics Association. doi:10.2312/stag.20211485. 10

[WTF09] WEISS Y., TORRALBA A., FERGUS R.: Spectral hashing. In *Advances in Neural Information Processing Systems 21*, Koller D., Schurmans D., Bengio Y., Bottou L., (Eds.). Curran Associates, Inc., 2009, pp. 1753–1760. 3