Enabling Hierarchical Exploration for Large-Scale Mutidimensional Data with Abstract Parallel Coordinates

Gaëlle Richer, Joris Sansen, Frédéric Lalanne, David Auber, Romain Bourqui
{gaelle.richer,joris.sansen,frederic.lalanne,david.auber,romain.bourqui}@u-bordeaux.fr
LaBRI, Univ. Bordeaux, CNRS, UMR5800
France

ABSTRACT

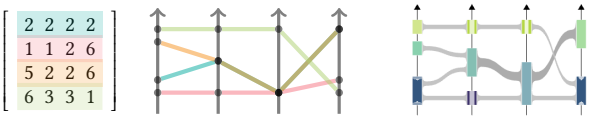
As data collection grows more common in various domains, there is a call for adapted or newer methods of visualization to tackle magnitudes exceeding the number of available pixels on screens and challenging interactivity. Exploratory visualization of large data present two major challenges: perceptual scalability and processing scalability. The first is concerned with overcoming the fundamental limitation of screens and human perception. The second deals with efficiently processing large volumes of data to achieve responsive interactions. Multiscale visualizations are an effective technique for solving the first challenge that builds on several levels of data abstraction to provide the user with an initial overview and subsequent incremental detail. The focus of this paper is on multidimensional data, a ubiquitous form of data among large-scale data sets, and parallel coordinates, a representation largely used for this type of data. For this representation, defining abstractions and interactively generating levels is not straightforward. Building upon several previous aggregated parallel coordinates representations, we propose a unifying and thinking model for conceiving and describing multiscale parallel coordinates and their interactions. Using this formalism, we present a focus+context representation which bounds the number of visual items with a fixed resolution parameter while supporting exploration up to the item-level. Processing scalability is addressed by carrying out computation in a distributed manner on a remote data-intensive infrastructure. Bounding the visual items ensures perceptual scalability but also bounds the data transfer between this infrastructure and the rendering client.

KEYWORDS

visualization, large-scale data, parallel coordinates, multiscale visualization, distributed computing, focus+context

1 INTRODUCTION

Multidimensional data, a form of data common in many domains, encompasses all lists of individuals composed of several attributes (possibly temporal). Multidimensional items are studied from several aspects: the particular behavior of individual items relative to the whole, the relationship between values from two dimensions and the distribution of values along each dimension [7]. Parallel coordinates, introduced by Inselberg and Dimsdale [19], is a well-known technique of visualization for such data. Each item is represented by a polyline which anchors on axes are positioned at the corresponding attribute value of the item (see Fig 1b). Axes are traditionally aligned in parallel forming a sequence of two-dimensional subplots sharing one axis with their predecessor. Usual interactions are *axis reordering* to analyze relationships between all dimension pairs and *selection* to trace subsets of items across axes. Selection relates to interactive means of choosing subsets of items and enhancing them such that they can be discriminated from the rest.



(a) Toy data (b) Line-based (c) Abstract

Figure 1: Examples of (b) line-based and (c) abstract parallel coordinate plots for the same dataset (a). On the line-based plot, colors match those of tuples on (a). On the abstract plot, the height of aggregates corresponds to the number of tuples they cover.

In this paper, we are interested in supporting the interactive exploration of multidimensional datasets with a *large* number of items and a moderate number of dimensions. We built upon the assumption that large datasets, scaling to billions of records, lead to major overplot when displayed using the traditional line-based parallel coordinate representation. Indeed, as the number of records increases, the plot becomes cluttered and analyses may be hindered as specific patterns are concealed [10]. Heinrich and Weiskopf [18] listed several clutter reduction approaches for parallel coordinates. One solution lies in the display of aggregates computed per-dimension as in [20, 29, 32] instead of single items (see Fig 1c). Despite the aggregation, these abstract plots successfully provide overview information similar to a traditional plot and perceptually scale for any size of input data. Rendering time and transfer time in client-server architectures are also reduced since they are bounded by the number of aggregates. Sansen et al. [32] leveraged this bound by precomputing the result of several interactions and thus clear the dependency of linear scans of the data upon interaction. Indeed, past a certain number of records, data cannot fit in a desktop computer memory and linear scans over the data affect performance more negatively since they either involve reads on disk or network transfer between several computing units.

As for any abstract visualization, deeper analyses are limited by the amount of information conveyed by the visual aggregates. Two types of interactions can alleviate this limitation: changing the level of detail (show *more* details) and adjusting the aggregation (show *different* details). Supporting these interactions is essential but strongly increases the number of states of the visualization and reasonable precomputing and storing of all these states as performed by Sansen et al. [32] is challenging. The processing scalability challenge here has been addressed by Rübel et al. [31] on a modern high-performance computing (HPC) platform. HPC platforms are large and expensive computing systems suited for highly complex and real-time computation. They are composed of multiple processors connected through a fast network and use fast memory. As such, they are particularly

adapted to *tightly coupled* tasks where several processors work on the same task and exchange data. Distributed systems, on the contrary, are networks of computing units, usually commodity hardware, connected in a *shared-nothing* architecture (memory and storage are independent to each unit). On these systems, data transfer between computing units uses a slower network connection and thus is critical for performances. Consequently, they are most adapted to *loosely coupled data-parallel* tasks on large amounts of data. In addition to being cheaper alternatives for data-intensive computing, they offer easier *horizontal scalability* (allocation of additional computing units) as their hardware and architecture are less sophisticated. The filtration and aggregation problems at stake in abstract parallel coordinates are data-parallel tasks. We focus on these less expensive and more accessible infrastructures to address them.

Our aim is to enable interactive exploration down to the item-level over large-scale multidimensional data with an abstract parallel coordinate plot. The main contributions of this paper are summarized as follows: (i) a unifying model for abstract parallel coordinate plots based on hierarchical aggregation, (ii) formalization of multiscale and regular interactions in this model, (iii) a focus+context representation with bounded visual items and drill-down/roll-up interactions, extension of the work of [32], (iv) an example implementation using a distributed infrastructure with components that benefits from horizontal scalability.

The paper is organized as follows: we first present previous work on perceptual scalability in parallel coordinate plots and their interactions with a focus on multiscale approaches (Sec. 2). Then, we describe the proposed graph-based formalism for abstract parallel coordinates based on hierarchical aggregation (Sec. 3). In Sec. 4, we study how different interactions are expressed in this model. This yields a prototype implementation and design of a scalable parallel coordinate plot based on hierarchical aggregation, using a so-called *big data* infrastructure. This visualization is described and its scalability evaluated in Sec. 5. Finally, in Sec. 6 we draw conclusions and present directions for future work.

2 RELATED WORK

Recent works have focused on the scalability of visualization applications for large-scale data with different techniques, among which are: data reduction, multi-threading, GPU-acceleration, and incremental or approximate data processing. In the case of visualization, the scalability of a system often refers to its capacity to accommodate and handle growing amounts of data. Handling massive datasets brings about two main challenges for exploratory visualization: *perceptual* scalability and *processing* scalability as noted by [12, 16, 24]. The first is concerned with the legibility of visualizations representing numerous items relative to the space available on a screen (so-called *screen real-estate problem*) and human capabilities to apprehend them. The second relates to the computational cost of processing numerous items on *each* user input, that can create latencies responsible for decreasing user performances [23]. A taxonomy of different techniques regarding the perceptual scalability aspect was established by Ellis and Dix [10]. A general solution is data reduction, which can be categorized into two approaches: either representing a subset of the data items (sampling, filtering) or meta-items (aggregation, mathematical models). Several work proposed methods (called multiresolution, multiscale, hierarchical or even stratified) for navigating through multiple levels of detail supported by precomputation (*e.g.* [15, 24, 30]). This work is related to general techniques addressing perceptual and processing scalability for interactive parallel coordinates.

2.1 Perceptual Scalability in Parallel Coordinates

Various approaches have been proposed to improve the legibility of parallel coordinate plots by either reducing the clutter produced by the multiplicity of overlapping and crossing lines or enhancing their patterns. Approaches can be categorized into *geometry-based* relying on computer graphics techniques and *data reduction* approaches that use approximation or summary of the data. Geometry-based approaches display all items with shape or position modifications to alleviate the overdraw in-between axes, for instance by bundling lines (*e.g.* [33]). These techniques have the advantage of producing no or few loss of information but have the drawback of still being prone to over-plot since no reduction of the number of displayed items is performed. Data reduction approaches limit the number of visual items either by sampling items [9] or using meta-items. Model-based approaches mathematically reduce the data to a continuous function, and meta-items usually represents the density of the underlying data (*e.g.* [27]). Aggregation approaches have been presented for parallel coordinates through different schemes: aggregation of dimensions (*e.g.* [2]), aggregation of items or values (*e.g.* [13, 28, 29]) or combinations of both (*e.g.* [21, 26]).

In this paper, we are interested in scalability relative to the number of items, not dimensions, hence we focus on aggregation over items and their values. Previous work using aggregation have used kernel density estimation, independently applied to dimension [29], different hierarchical clustering algorithms over multidimensional items or dimension (*e.g.* [13]) but also binning on two-dimensional sub-spaces (*e.g.* [14, 28]). Aggregates have been represented by their statistical properties: extrema (*e.g.* [3]), cardinality (*e.g.* [20]), mean (*e.g.* [13]), and/or other metrics [29].

As reflected by previous work, multiple dimensions allow for different levels of grouping: the item level (multidimensional), the value level (per-dimension), the line level (two-dimensional). For aggregation-based methods, the value level has the advantages of losing less information, preserving continuity on axes (potentially broken by two-dimensional aggregation) as well as an idea of the relationship between dimensions (broken by item-level aggregation).

2.2 Multiscale Parallel Coordinates

Abstract representations are usually conditioned by a parameter that controls the resolution of the display. Multiscale visualizations allow navigation between multiple levels of abstraction, using a *drill-dow*/*roll-up* interaction (respectively for increasing and decreasing levels of detail). Elmqvist and Fekete [11] provided general guidelines for multiscale representations and interactions, based on such hierarchical aggregation. Bikakis et al. [4] presented a framework for hierarchical aggregation oriented towards the computational aspects of hierarchical navigation. Interactively changing the level of detail can be integrated in different ways (see [8] for a general review). A first approach, similar to geometry zooming, either (1) filters part of the representation to maintain a fixed number of visual items on screen or instead (2) purposely displays increasing number of items (*e.g.* [6, 29]). Filtering has the advantage of being scalable when the amount of displayed items is controlled; however, the overview and context

are lost along the way. One drawback of displaying an uncontrolled number of items is that the screen pixel limitation could always be reached for a sufficiently massive dataset and information would be lost. Additionally, from a certain point it may overwhelm human capacities. Consequently, zooming without filtering does not allow interactive exploration to the item-level complying with perceptual scalability. For zooming with filtering, an alternative to the loss of context is the *overview+detail* approach which displays both the filtered zoomed view and an overview (*e.g.* dimension zooming of [13]).

A second approach, called *focus+context*, consists in displaying heterogeneous levels of detail: a selected portion of the data is shown with greater details to the detriment of the rest. This approach allows to gain detail locally while preserving the overview. Fisheye lenses used in the work of [25] are an example applied in screen space. In data space, [13] and [28], although using different aggregation strategies, presented techniques where a subset of items can be enhanced and displayed with fine-scale details layered over the rest of the data, abstracted to some level.

To the best of our knowledge, none of these solutions propose to interactively change the level of detail locally, in a focus+context fashion, while bounding the number of visual items.

2.3 Processing Scalability of Multiscale Parallel Coordinates

For the past ten years, *large-scale* or *massive* has been used to qualify increasingly big data, now up to tera or petabytes in size [12]. For multidimensional data, above about 10^7 items, with a dozen of dimensions, perceptual scalability and processing scalability becomes problematic for interactive analysis. Abstract representation and precomputation of interactions are solutions respectively addressing perceptual and processing scalability, as [32] presented. In addition to the enhancement of subsets of items and the reordering of axes, abstract representations require a drilling interaction to allow fine data analysis.

Processing scalability can be tackled by parallelism, distributed processing and precomputing of complete or partial results for instance. For parallel coordinates, most interactions can be applied to two-dimensional subplots independently and on separate portions of the dataset without needing any communication (they are *pleasantly parallel*). This property has been exploited by Rübel et al. [31] on a HPC platform, and [32] on a distributed platform. Rübel et al. [31] addressed both scalability challenges with an histogram-based representation adapted from [28]. Histograms are two-dimensional aggregations of the data, precomputed in parallel, potentially for different resolutions. On more affordable hardware, Sansen et al. [32] addressed the same challenges with a parallel sets representation for an Hadoop+Elasticsearch ecosystem that also relies on full precomputation of certain types of interactions. Both works presented good scalability evaluations relative to the number of computing units used. However, these techniques only support the display of balanced levels of detail *i.e.* drill-down is *global* and necessarily increases the number of displayed items. With this approach, gaining detail at the item-level does not scale to large datasets.

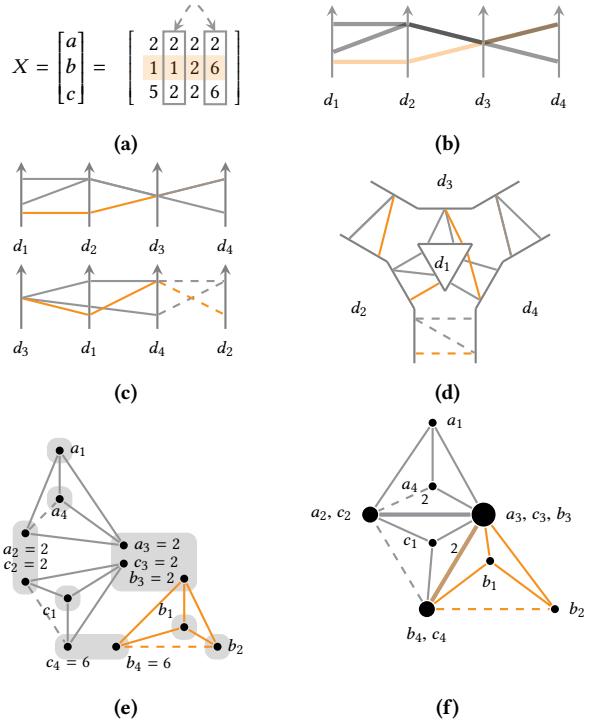


Figure 2: Rationale behind the graph model. (a) Example data. (b) Opacity-based parallel coordinates. (c) Parallel coordinates matrix where all 2-dimensional subspaces are represented. (d) Many-to-many plot. (e) Graph-based representation $G(X)$. Shadowed areas cover identical values from the same dimension and form the partition R. (f) Quotient graph $G_{/R}(X)$ (cardinality mapped to size).

3 A GRAPH-BASED FORMALISM FOR ABSTRACT PARALLEL COORDINATES

Consider $X \in \mathbb{R}^{n,d}$, a set of multidimensional quantitative data with n items and d dimensions. On a traditional parallel coordinate plot, only a fraction of the input data subspaces is represented. Fig. 2c and 2d show two examples of representations that display all subspaces at once: respectively the parallel coordinate matrix [17] and the many-to-many plot [22]. The model we propose integrates all subspaces without assumption on the layout of axes and supports abstraction. A value in the matrix inherently belongs to a tuple (or item), and a dimension. We focus on abstraction based on aggregation of tuples and do not directly address aggregation of dimensions. The rationale behind this approach is that dimensions hold semantic meaning not embedded in their numerical content. Therefore, automatically and meaningfully aggregate them is not straightforward: some types of values may not make sense aggregated together. Consequently, we target datasets that challenge scalability by their size in tuples (more than millions of items) and hold moderate amounts of well-chosen dimensions.

3.1 Abstraction in Parallel Coordinates

In parallel coordinates, the values of the input matrix are connected to each other by lines to materialize the tuple or tuples they belong to. Extending this metaphor, an $n \times d$ matrix can be seen a graph linking its values based on the relation induced by tuples. This undirected graph is composed of n separate cliques

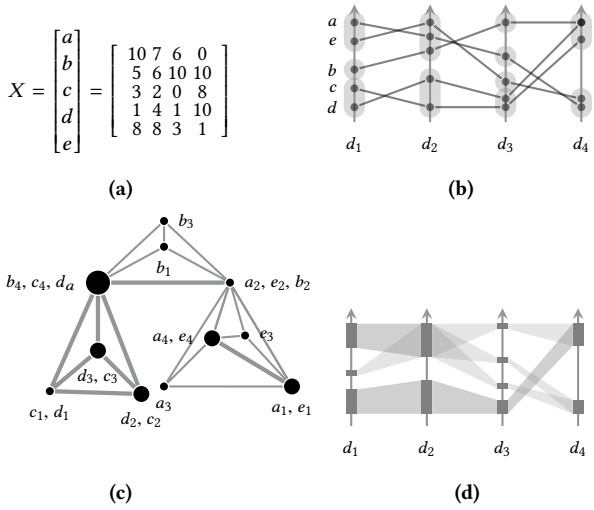


Figure 3: Aggregation of dimension values. (a) Example data (b) Parallel coordinate plot augmented by per-dimension partitions. (c) Quotient graph. (d) Value-oriented abstract representation using extrema values and cardinality (with link opacity).

of d vertices, that is, n complete graphs (see Fig. 2e). We note this graph $G = (V, E, w)$, where V denotes the vertices, E the edges, w is the weighting function: $w : V \to \mathbb{R}$ that stores vertex values. We note $D = \{V_i, i \in [\![1, d]\!]\}$ the partition of V that groups vertices along their origin dimension. Given a dimension sequence, a parallel coordinate plot is built by taking the subgraph of G composed of only the edges joining vertices of consecutive dimensions in the sequence. Vertices of this subgraph are positioned on their corresponding axis along a vertical position given by the weighting function. Shadowed regions on Fig. 2e represent a vertex partition R that groups identical values of the same dimension. Such partition is a *refinement* of the partition D. Merging vertices of G belonging to the same subset of a partition R corresponds to taking the *quotient graph* of G relative to R, noted $G_{/R}$. Two subsets S, S' in the partition R are represented by adjacent vertices in $G_{/R}$ if, and only if, some vertex in S is adjacent to a vertex in S' in G. Further, two aggregation functions are defined to compute the properties of the *meta-nodes* and *meta-edges*, i.e. vertices and edges of the quotient graph. Aggregation functions return a tuple of weights given a set of edges or vertices as input. Weights (cardinality, extrema, mean, etc) provided by such functions are subsequently used for assigning visual properties to meta-edges and meta-nodes. For instance, on Fig. 2b, the opacity of lines on the parallel coordinate plot depends on the cardinality of meta-edges. On this example, the aggregation of identical values allows to find the optimal number of lines to draw. Indeed, the aggregation done here in data-space relates to the one done in screen-space when rendering the polylines with alpha composing.

Given some aggregation functions, the same process is applied to obtain an abstract representation from any *valid* partition R of V, that is, refinement of D. Fig. 3 presents an example of partition obtained with per-dimension clustering and another type of representation based on extrema values. Tuple-based approaches are incorporated into the model by refining the input tuple partition into a refinement of D. Each subset of the tuple partition can be decomposed into d groups along their membership in

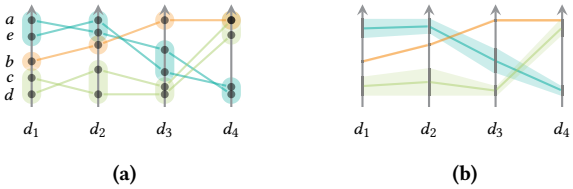


Figure 4: Aggregation of tuples. (a) Translation of the tuple partition (color) into a vertex partition refining D (shadowed regions). (b) Tuple-oriented abstract representation using mean and extrema values.

D. Fig. 4 shows an example of tuple partition refinement and a tuple-oriented representation. Using our model and the introduced notations, the abstract representation is fully defined by: (i) a valid partition of the matrix values, (ii) two aggregate functions assigning weights to meta-nodes and meta-edges, (iii) an axis layout and visual encodings for meta-node and meta-edge weights (vertical position, color, etc).

3.2 Hierarchical Abstraction in Parallel Coordinates

Hierarchical aggregations are precomputation of different levels of abstraction that naturally support multiscale representations. Such precomputation outputs a rooted tree structure whose leaves are the data objects to aggregate. Every node of a rooted tree is said to *cover* the leaves of the subtree it is the root of. An *antichain* (also called *tree cut*) in a tree is a set of nodes S, such that no node of S is an ancestor of another node of S. An antichain is *maximal* if it cannot be expanded by any other node without violating the antichain property. The subsets covered by the nodes of a maximal antichain form a partition of the tree leaves: the antichain property makes them pairwise disjoint, the maximal property ensures that their union is the set of leaves.

In our model, an abstraction is directed by a partition refining the partition D which is equivalent to d partitions, one for each V_i of D. By augmenting each subset V_i with a tree, such partition can be defined by one maximal antichain for each tree. This is equivalent to defining a single maximal and non-trivial antichain in the unified tree where the *root*'s children are the roots of all dimension trees. We note this unified tree $T(V)$, and call its direct subtrees *dimension hierarchies*.

Overall, for some input data modeled by the clique graph $G = (V, E, w)$ and its associated hierarchy $T(V)$, an abstraction is defined as previously described with the valid partition of V induced by a maximal antichain in $T(V)$. This unifying approach accommodates multiple layouts of parallel coordinates (see Fig. 2) and several abstract parallel coordinates representations. We presented two abstract representations: per-dimension aggregation on Fig. 3d that corresponds to the technique presented by Palmas et al. [29] and on Fig. 4 tuple aggregation like presented by Fua et al. [13]. For tuple aggregation, the hierarchy defined on tuples is used to form all dimension hierarchies. Additionally, to represent the same tuple aggregates on all subplots, dimension hierarchies should all be cut at the same level.

4 COMMON INTERACTIONS

We take advantage of the hierarchical graph model described in the previous section to explore several hierarchical operations

and usual operations for parallel coordinates and investigate the incremental computation they require.

4.1 Level-of-Detail Operations

The *drill-down* and *roll-up* operations correspond to changes in the displayed level of detail. They respectively mean moving deeper in the hierarchical aggregation to show finer details, and upper to present a coarser display of the data. Since the model rests upon precomputed hierarchies of dimension values, the weights (cardinality, extrema, etc) of all possible meta-nodes, *i.e.* nodes of the hierarchies, can be precomputed as well. The number of edges of a quotient graph with respect to a maximal antichain is always less than those of the original graph and generally greatly depends on the properties of the hierarchies (arity, depth). In general, the number of maximal antichains is exponential. For example, *complete* binary trees, that are rooted binary trees with every level full except the last, have $\Omega(2^n)$ maximal antichains where n is their number of leaves. This makes the precomputation of all possible quotient graphs, *i.e.* all meta-edge weights, with respect to every possible maximal antichain for given dimension hierarchies not possible for the scale of data we aim to tackle.

In traditional parallel coordinates plots, and in many derived techniques, each two-dimensional subplot is independent of the others and can be computed and displayed separately. When axes hold visual elements, like frequency plots, the corresponding visual items are usually mirrored and the two composing parts conceptually included in both sides subplots. If no edge is precomputed, the cost of an abstraction can be decomposed into independent substeps corresponding to all subplots, each requiring aggregation over n edges, one for each tuple.

In the context of hierarchical aggregation, gaining detail can be defined locally as the substitution of a visual aggregate for its children. In our model, the displayed level of detail is given by a maximal antichain of the hierarchy. *Drilling-down* on a meta-node replaces it in the current antichain by all of its children which still forms a maximal antichain of the tree. This entails computing the outgoing edges of these children since they are not precomputed. It corresponds to aggregating the tuples covered by the drilled meta-node. The opposite operation, rolling up or *collapsing*, is the replacement of sibling nodes with their parent in the antichain.

Globally increasing detail may result in an uncontrolled number of elements on display, independently of the arity and depth of hierarchies. We detail three drilling approaches that tackle this issue by trading context detail for scalability.

Detail & Filter. One approach is to define drilling on a meta-node as a filtering operation. This corresponds to computing the expansion of the meta-node over a filtered clique-graph, induced solely by vertices adjacent to the leaves it covers in the original clique-graph. This method effectively bounds the number of aggregates on an axis by the arity of the dimension hierarchy, *i.e.* the maximal number of siblings in the tree. This also bounds the number of edges in an abstract plot. The limit of this approach is that all subplots are modified since the weights of all meta-nodes and meta-edges from the quotient graph are to be updated to account for the filtering. Thus for each subplot, it costs a pass over as many bottom-level edges as the drilled meta-node covers items.

Budgeted Detail. A second approach lies in constraining the definition of maximal antichains such that their size comply with a predefined budget. In this setting, drilling potentially implies modifying the current maximal antichain in multiple points such that previously acquired detail collapses to allow drilling-down when the budget would have been exceeded otherwise. If these changes are restricted to a single dimension hierarchy, the incremental changes only involve one or two subplots. Computing these subplots costs a pass over n bottom-level edges for each, in addition to the cost of finding a suitable maximal antichain.

For a budget defined in number of visual items, the minimal budget allowing to define a maximal antichain that does contain a leaf node depends on the arity and depth of the tree. In a tree T with n leaves and arity a, a maximal antichain containing at least one leaf and minimal in size can be expected to have between $depth(T)$ and $a \cdot depth(T)$ nodes. Considering the toy example of binary trees which nodes all have either 0 or 2 children, the minimum depth is $\lceil log_2(n) \rceil + 1$ and the maximal depth is n. Consequently, a visual budget allowing gaining detail up to the leaf level has to be chosen with respect to the depth of the hierarchy. In our context, since this depth can reach orders of magnitude same as the number of tuples, this method does not scale without strong constraints on the properties of hierarchies.

Dynamic Context Aggregation. An alternative solution to collapsing nodes when focusing on deeper nodes is to aggregate them dynamically. The meta-nodes from the chosen antichain that are not in focus (*i.e.* not the deepest in the hierarchy) are represented aggregated which lowers their impact on the visual item budget. In general, these dynamically-computed aggregates, do not correspond to existing nodes in the precomputed hierarchy. This approach limits the visual items on an axis to the siblings of each focus and the aggregates of the rest of the meta-nodes that forms the context. Thus, for a dimension hierarchy of arity k, an axis with f foci corresponds to at most $k \cdot f + (f + 1)$ visual nodes, whatever the depth of these foci in the hierarchy is. Forming these aggregates induces an aggregation of their outgoing edges, therefore it similarly reduces the number of meta-edges. Consequently, with additional constraints on the arity of dimension hierarchies and the number of concurrent foci on an axis, this approach enables drill-down up to the deepest level while complying with a visual budget. Contrary to the previously described approaches, *dynamic context aggregation* matches the requirement of our system and thus is the one implemented (see Sec. 5.1 and 5.2).

4.2 Other Operations

Axis Reordering can be conceived either as the computation of a full plot or, incrementally, as an edition of one or more subplots which requires one or two passes over n bottom-level edges, for n the number of tuples. *Subset Selection* relates to the emphasis of a subset of tuples in a given representation. In an aggregate visualization setting, an abstraction can be computed over the subset of tuples and the resulting weights displayed over the complete abstraction using a discriminating encoding. Using this approach, a subset selection of m tuples requires going through m bottom-level edges per subplot, one per tuple, to compute their contribution to the current meta-edge weights. The process is the same for meta-node weights.

5 FOCUS+CONTEXT REPRESENTATION & SCALABLE IMPLEMENTATION

Our goal is to enable hierarchical exploration in abstract parallel coordinates while complying with the following scaling properties. On the representation-side, exploration should be possible down to the item level, in a top-down manner, while the number of visual items on display should be bounded for any size of input data. On the processing-side, network transfer latency between the displaying unit and the computing unit should be bounded and traditional operations (axis reordering, subset selection) should be supported. In the following, we present first the cornerstone of our method: the budgeted number of visual items. Then, we detail a novel focus+context display with intuitive drill-down capabilities using the *dynamic context aggregation*. And finally, we describe technical details and performance evaluation of the proposed system.

5.1 Bounded Number of Visual Items

Bounding visual items is essential for perceptual scalability but also ensures that data transfer between our rendering client and back-end unit remains bounded in size thus predictable in time. To this end, the properties of dimension hierarchies are constrained and the number of foci restricted to one per dimension.

Hierarchy Constraints. We use a user-defined k value that acts as a *resolution parameter*, bounding the number of visual items per displayed axis. This parameter is enforced as the maximal arity of dimension hierarchies. k should be chosen large enough for the representation to preserve an appropriate amount of information but small enough for the visual items to fit the available screen space. Additionally, dimension hierarchies should order their leaves with respect to their values such that every maximal antichain defines partitions of dimension *intervals* since the chosen visual representation relies on this property. Binning (equal range partitioning) and adaptive binning (equal size partitioning) are examples of partitioning algorithms that can be applied in a bottom-up fashion to produce hierarchies complying with these two requirements. No depth constraint is required, we adopt the dynamic context aggregation strategy to bound the number of vertices from quotient graphs displayed.

Level-of-Detail Navigation. We define as *focus nodes*, the nodes that have the maximal depth in the current antichain. The rest of the antichain nodes from one dimension are aggregated into the minimal number of *context nodes* such that their order is preserved. In the proposed implementation, the top-level nodes are initially presented as focus nodes and each drill-down triggers a dynamic aggregation (examples are presented in Fig. 5). Consequently, there is no more than k focus nodes and two context nodes at once per axis. Thus, the number of nodes on display is bounded by $k + 2$ per axis and the number of edges by $(k + 2)^2$ per subplot.

5.2 Visual Encoding

We propose an extension of the work presented in [32] which included two visual encodings: one oriented towards the distribution of tuples, the other towards the distribution of values themselves. Basically, the first encoding maps aggregate height to their cardinality, while the second maps it to their covered interval size (distance between extrema). We extend the encoding and positioning of aggregates to emphasize the *focus regions*, composed on each dimension of the data displayed with the finest

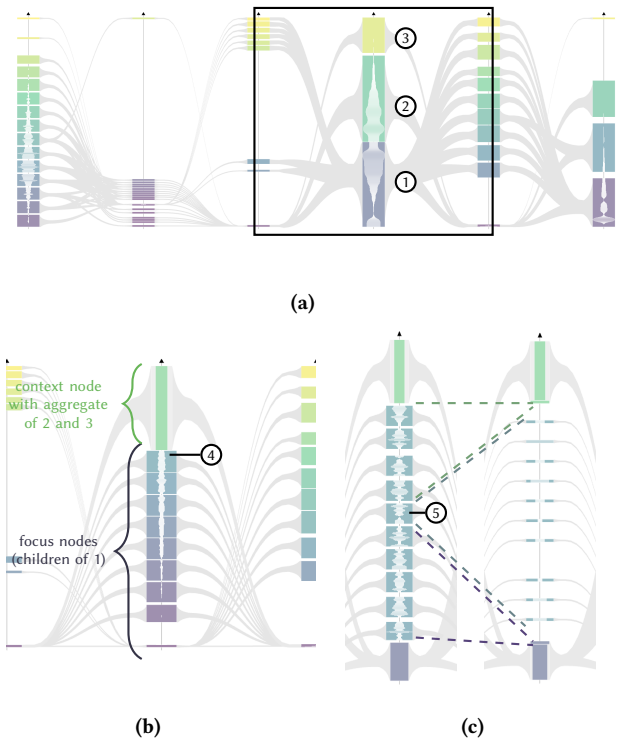


Figure 5: Focus+context representation. (a) Initial display. (b) Drill-down on ①. The top context node represents the aggregation of ② and ③. (c) Drill-down on ④ and ⑤ successively.

detail. Context regions are displayed with slightly lower width to emphasize the focus regions. Since regions of interest are incrementally refined by successive drill-down, they get smaller in height relative to the whole both in terms of cardinality and interval of values. Therefore, the height of focus and context regions are rendered with different scaling factors. Focus nodes are represented colored and augmented by a bidirectional smoothed histogram. Context nodes are augmented by a pile of *level blocks* (see Fig. 5c). The wider these blocks are, the lower in the hierarchy are the nodes they represent (notice on Fig. 5c the difference in width of the outer blocks from the top and bottom context). The height and vertical order of level blocks follows the same encoding as those of focus nodes. Level blocks are computed on the client side and thus not transferred nor counted in the visual item budget.

In the drilling mode, focus nodes and level blocks are clickable. Clicking on a focus node triggers its animated expansion which split it into its children and merges its siblings into level blocks. Level blocks represent an aggregation of nodes previously in focus, thus they enable going back to this precise previous state.

5.3 System Overview

Our system consists of two main parts and follows a client/server architecture where the client is the visualization endpoint and the server is an interface to an on-demand computing and preprocessing back-end. The latter could be implemented both in a distributed environment for a Hadoop cluster and as a multi-threaded application for single machines (desktop computer or dedicated server). Past a certain number of input records (for a

given number of dimensions and resolution parameter), a distributed platform should be more efficient while facilitating load expansion.

The client is a WebGL application that displays the representation, computes level blocks and context aggregation on node drill-down, and queries the supporting back-end for other interactions. The back-end server is a long-lived Spark application which runs distributed job on demand while keeping prepared data in memory. It computes dimension hierarchies in a precomputation step and stores the resulting meta-node weights (extrema and cardinality) for all the hierarchies in a distributed database. Contrary to the previous system, the hierarchical aspect makes the number of displayable meta-edges too large to allow their precomputation in reasonable time (see Sec. 4.1).

The membership of each input data value is stored in a *hierarchy matrix* of the same size as the input data, where each value holds the list of computed ancestors for the matching input data value. This matrix is kept in memory and split among computing units which will pass over their slice of the data to filter and aggregate results on demand. The aggregation outputs the meta-edges (source, target and cardinality) as well as the weights of context nodes since they do not exist in the precomputed hierarchies. Upon user interaction and if necessary, the client requests the server which in turn runs a distributed operation and merges the partial results returned by computing units. Finally, the client receives the incremental changes in plain text and updates the view consequently.

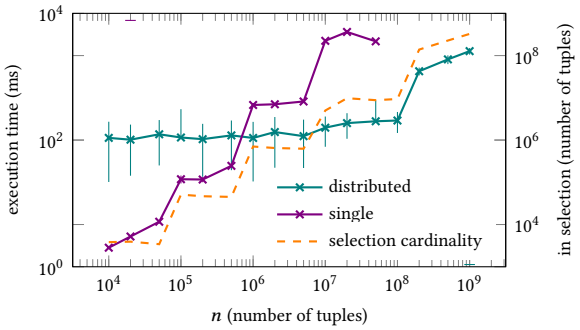
5.4 Performance Evaluation

Our system supports moving an axis, selecting an aggregate node or edge, drill-down on a focus node and rolling-up by clicking on a context part. Drill-down, roll-up and moving an axis correspond to the same low-level operation which consists in computing the edges of two-dimensional subplots given the current focus on each dimension involved. Compared to the other operations that only modify two to three subplots, selection operations affect all subplots, thus are more computationally expensive.

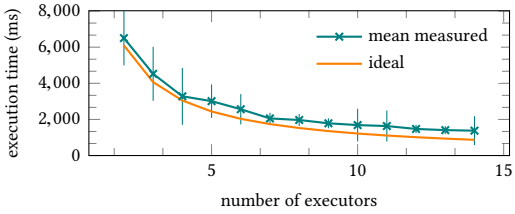
Since we are interested in supporting "$n \gg d$" datasets, we evaluate our system for varying number of tuples n and using the most expensive interactive operation: node selection of the largest node (in covered tuples). To demonstrate the scalability of the approach we also evaluate performance relative to the resources allocated for computation.

Each node of the distributed platform has 64GB RAM and 2x6 hyper-threaded cores at 2.10GHz each, connected via a 1Gbit/s network. The single computer used for running the multi-threaded implementation has 2x4 hyper-threaded cores at 3.3GHz and 64GB RAM. Test datasets are generated for varying n (from 10^4 to 10^9), with $d = 15$ and $k = 31$. Test datasets are generated such that pairs of dimensions present a close to null correlation factor [5] which tends to create close to the maximum number of edges between dimensions. Dimension hierarchies are generated using Canopy clustering [1] applied in a bottom-up manner.

On Fig. 6a, execution times of the selection of the largest top-level node are plotted for increasingly large datasets. We evaluate both a multi-threaded single-computer implementation and the distributed Spark implementation. The stairs-shaped curve of the single computer can be explained by the similar trend of the selected node cardinality for every test dataset. Indeed, due to the hierarchy constraints and the bottom-up approach for clustering, the number of nodes on the top-level varies and the cardinality



(a) Execution time for node selection relative to data size for two implementations. The distributed version uses 15 executors. Errors bars are not symmetrical and missing their bottom parts on two points due to the log-log scale.



(b) Scalability test using node selection, $n = 2 \cdot 10^8$.

Figure 6: Performance evaluations. Experimental results are averaged over 1000 runs. Executor resources (12 cores, 31GB memory) are sized to match those of physical units.

of the largest nodes does not increase linearly with n. We observe that, on our infrastructure, under $n = 2 \cdot 10^8$ (about 10^8 selected tuples) the Spark application performances are stable despite increasing workload. This suggests that execution time for datasets of smaller size is dominated by costs related to network and disk I/O, and/or by merging all executor results by the *driver* unit. Indeed, the cost of merging results remains approximately the same as it is a function of the output size and the number of task running in parallel. Another visible aspect of the results on small datasets is their high variation. The oversized allocated memory can be an explanation: *when* triggered, garbage collection incurs a substantial delay. Overall, for n less than approximately 10^6, the distributed infrastructure underperforms the single-computer despite having more resources. Past this limit, the system is better tuned and seems to scale linearly relative to the number of tuples in the selection. This observation holds for the other experiments carried out for edge selection, roll-up, and drill-down.

We investigate the scalability of the system using a dataset larger than this limit. Fig. 6b presents the median execution time for node selection on a $2 \cdot 10^8$-tuple dataset with varying numbers of executors. The plotted *ideal* execution time corresponds to a linear speedup, that is, halving the execution time when doubling the number of executors. This experiment shows that the system performs close to the ideal.

6 CONCLUSION

We presented a graph model for hierarchical aggregation and interaction strategies for parallel-coordinate-based visualization. This model formalizes aggregation over multidimensional data at

the most-expressive level by making use of per-dimension hierarchies. This approach treats all dimensions equivalently which matches the way dimensions are handled in parallel coordinates.

Based on this model, we presented a client/server system for interactive exploration of large multidimensional data using abstract parallel coordinates. We address the limitation of the previous system, based on fixed partitions of dimension values, with hierarchies that allow to interactively change partition. The proposed drill-down operation enables the definition of an arbitrary-detailed focus region on each axis while retaining context in a reduced form. On the client-side, we showed how to display focus and context regions with intuitive navigation cues. The strength of this approach is that it supports exploration down to the item-level while controlling the number of visual items, thus ensuring perceptual scalability.

On the server-side, the back-end processing is handled on a distributed platform with components that scale horizontally. Besides drill-down/roll-up operations, the implementation supports standard parallel coordinate operations. Experimental results demonstrate the scalability of the back-end system relative to the size of the input data and to the resources allocated for computation. The results indicate that the proposed system can support increasingly large datasets by expanding its network of computing units. To a certain extent, adding computing resources can also reduce interaction latencies.

One focus of the design is the bounded data transfer between the client and server parts which relies on the single-focus approach (on each axis) and the precomputation of dimension hierarchies with a bounded arity k of small orders of magnitude. The hierarchies being predefined allows for efficient filtering and aggregation of items based on ancestors but may be limiting for exploration. As a counterbalancing measure, future work could add support for user-driven hierarchy refinement. Depending on the scope and scale of the refinements this could be handled on the client-side and/or enforced on the server-side.

Another path for future work is methods for latency reduction other than horizontal scaling. Space is a possible trade-off. Even if the number of possible meta-edges makes their total precomputation infeasible, partial precomputation could be investigated: either beforehand or as a background process targeting meta-edges that are the most likely to be requested given the current state.

ACKNOWLEDGMENTS

This work has been carried out as part of the "REQUEST" (PIAO18062-645401) and "SpeedData" (PIAO17298-398711) projects supported by the French "Investissement d'Avenir" Program (Big Data – Cloud Computing topic). The authors would also like to thank the anonymous referees for their valuable comments and helpful suggestions.

REFERENCES

[1] A.McCallum, K. Nigam, and L. H. Ungar. 2000. Efficient clustering of high-dimensional data sets with application to reference matching. In *Knowledge Discovery and Data Mining*. ACM, 169–178.

[2] K. Andrews, M. Osmic, and G. Schagerl. 2015. Aggregated parallel coordinates: integrating hierarchical dimensions into parallel coordinates visualisations. In *Proc. of Int. Conf. on Knowledge Technologies and Data-driven Business*. ACM.

[3] G. Andrienko and N. Andrienko. 2004. Parallel coordinates for exploring properties of subsets. In *Proc. of Int. Conf. on Coordinated & Multiple Views in Exploratory Visualization*. 93–104.

[4] N. Bikakis, G. Papastefanatos, M. Skourla, and T. Sellis. 2017. A hierarchical aggregation framework for efficient multilevel visual exploration and analysis. *Semantic Web* 8, 1 (2017).

[5] S. Börzsönyi, D. Kossmann, and K. Stocker. 2001. The Skyline Operator. In *Proc. of Int. Conf. on Data Engineering*. IEEE Computer Society, 421–430.

[6] K. Selçuk Candan, L. Di Caro, and M. L. Sapino. 2012. PhC: Multiresolution Visualization and Exploration of Text Corpora with Parallel Hierarchical Coordinates. *ACM TIST* 3, 2 (2012).

[7] J. H. T. Claessen and J. J. van Wijk. 2011. Flexible Linked Axes for Multivariate Data Visualization. *IEEE Trans. Vis. Comput. Graph.* 17, 12 (2011).

[8] A. Cockburn, A. K. Karlson, and B. B. Bederson. 2008. A review of overview+detail, zooming, and focus+context interfaces. *Comput. Surveys* 41, 1 (2008).

[9] G. Ellis, E. Bertini, and A. Dix. 2005. The sampling lens. In *CHI '05*. ACM Press.

[10] G. P. Ellis and A. J. Dix. 2007. A Taxonomy of Clutter Reduction for Information Visualisation. *IEEE Trans. Vis. Comput. Graph.* 13, 6 (2007).

[11] N. Elmqvist and J.-D. Fekete. 2010. Hierarchical Aggregation for Information Visualization: Overview, Techniques, and Design Guidelines. *IEEE Trans. Vis. Comput. Graph.* 16, 3 (2010).

[12] D. Fisher. 2016. Big data exploration requires collaboration between visualization and data infrastructures. In *Proc. Workshop on Human-in-the-Loop Data Analytics*.

[13] Y.-H. Fua, M.O. Ward, and E.A. Rundensteiner. 1999. Hierarchical parallel coordinates for exploration of large datasets. In *Proc. Visualization'99*. IEEE.

[14] Z. Geng, Z. Peng, R. S. Laramee, J. C. Roberts, and R. Walker. 2011. Angular Histograms: Frequency-Based Visualizations for Large, High Dimensional Data. *IEEE Trans. Vis. Comput. Graph.* 17, 12 (2011).

[15] P. Godfrey, , J. Gryz, P. Lasek, and N. Razavi. 2016. *Interactive Visualization of Big Data*. Springer International Publishing, 3–22.

[16] P. Godfrey, J. Gryz, and P. Lasek. 2016. Interactive Visualization of Large Data Sets. *IEEE Trans. Knowl. Data Eng.* 28, 8 (2016).

[17] J. Heinrich, J. Stasko, and D. Weiskopf. 2012. The Parallel Coordinates Matrix. *EuroVis–Short Papers* (2012).

[18] J. Heinrich and D. Weiskopf. 2013. State of the Art of Parallel Coordinates. In *Eurographics 2013*. Eurographics Association.

[19] A. Inselberg and B. Dimsdale. 1990. Parallel Coordinates: A Tool for Visualizing Multi-dimensional Geometry. In *IEEE Visualization*.

[20] R. Kosara, F. Bendix, and H. Hauser. 2006. Parallel Sets: Interactive Exploration and Visual Analysis of Categorical Data. *IEEE Trans. Vis. Comput. Graph.* 12, 4 (2006).

[21] A. Lex, M. Streit, C. Partl, K. Kashofer, and D. Schmalstieg. 2010. Comparative Analysis of Multidimensional, Quantitative Data. *IEEE Trans. Vis. Comput. Graph.* 16, 6 (2010).

[22] M. Lind, J. Johansson, and M. D. Cooper. 2009. Many-to-Many Relational Parallel Coordinates Displays. In *IV 2009*. IEEE Computer Society.

[23] Z. Liu and J. Heer. 2014. The Effects of Interactive Latency on Exploratory Visual Analysis. *IEEE Trans. Vis. Comput. Graph.* 20, 12 (2014).

[24] Z. Liu, B. Jiang, and J. Heer. 2013. *imMens*: Real-time Visual Querying of Big Data. *Comput. Graph. Forum* 32, 3 (2013).

[25] T. V. Long and L. Linsen. 2009. MultiClusterTree: Interactive Visual Exploration of Hierarchical Clusters in Multidimensional Multivariate Data. *Comput. Graph. Forum* 28, 3 (2009).

[26] J. Lu, Y. Zhang, J. Xu, G. Xiao, and Q. Liang. 2014. Data Visualization of Web Service with Parallel Coordinates and NodeTrix. In *SCC 2014*. IEEE Computer Society.

[27] H. Nguyen and P. Rosen. 2017. DSPCP: A Data Scalable Approach for Identifying Relationships in Parallel Coordinates. *IEEE Trans. Vis. Comput. Graph.* (2017).

[28] M. Novotny and H. Hauser. 2006. Outlier-Preserving Focus+Context Visualization in Parallel Coordinates. *IEEE Trans. Vis. Comput. Graph.* 12, 5 (2006).

[29] G. Palmas, M. Bachynskyi, A. Oulasvirta, H. P. Seidel, and T. Weinkauf. 2014. An Edge-Bundling Layout for Interactive Parallel Coordinates. In *2014 IEEE Pacific Vis. Symp.* IEEE.

[30] A. Perrot, R. Bourqui, N. Hanusse, F. Lalanne, and D. Auber. 2015. Large interactive visualization of density functions on big data infrastructure. In *Symp. on Large Data Analysis and Visualization*. IEEE.

[31] O. Rübel, Prabhat, K. Wu, H. Childs, J. S. Meredith, C. G. R. Geddes, E. Cormier-Michel, S. Ahern, G. H. Weber, P. Messmer, H. Hagen, B. Hamann, and E. W. Bethel. 2008. High performance multivariate visual data exploration for extremely large data. In *SC '08*.

[32] J. Sansen, G. Richer, T. Jourde, F. Lalanne, D. Auber, and R. Bourqui. 2017. Visual Exploration of Large Multidimensional Data Using Parallel Coordinates on Big Data Infrastructure. *Informatics* 4, 3 (2017).

[33] J. Wang, X. Liu, H.-W. Shen, and Guang Lin. 2017. Multi-Resolution Climate Ensemble Parameter Analysis with Nested Parallel Coordinates Plots. *IEEE Trans. Vis. Comput. Graph.* 23, 1 (2017).