Importance-Driven Visualization Layouts for Large Time Series Data

Ming C. Hao, Umeshwar Dayal[1]
Hewlett-Packard Laboratories, Palo Alto, CA

Daniel A. Keim, Tobias Schreck[2]
University of Konstanz, Germany

ABSTRACT

Time series are an important type of data with applications in virtually every aspect of the real world. Often a large number of time series have to be monitored and analyzed in parallel. Sets of time series may show intrinsic hierarchical relationships and varying degrees of importance among the individual time series. Effective techniques for visually analyzing large sets of time series should encode the relative importance and hierarchical ordering of the time series data by size and position, and should also provide a high degree of regularity in order to support comparability by the analyst.

In this paper, we present a framework for visualizing large sets of time series. Based on the notion of inter time series importance relationships, we define a set of objective functions that space-filling layout schemes for time series data should obey. We develop an efficient algorithm addressing the identified problems by generating layouts that reflect hierarchy- and importance-based relationships in a regular layout with favorable aspect ratios. We apply our technique to a number of real-world data sets including sales and stock data, and we compare our technique with an aspect ratio aware variant of the well-known TreeMap algorithm. The examples show the advantages and practical usefulness of our layout algorithm.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphics]: Picture/Image Generation – Display Algorithms; H.5.0 [Information Systems]: Information Interfaces and Presentation – General.

Additional Keywords: Information Visualization; Time Series; Space-Filling Layout Generation.

1 INTRODUCTION

Time series are a data type of utmost importance in many application domains. Information Visualization to date has contributed with a variety of helpful techniques to understand and analyze time series data, where the focus has been mainly to support a limited number of time series, or to consider aggregated views of large collections of time series. The Polaris [9] system, for example, allows the analyst to easily pivot and refine visual specifications of table-based graphical displays. Schumann [1] employs a time wheel, where the basic idea is to present the time axis in the center of the display, and circularly arrange the variables around the time axis. Van Wijk [14] introduced a clustering-based visualization to condense multiple time series data into a calendar-based view. Shneiderman's interactive pattern search [3] provides fast information retrieval on over-laid time series data. Sets of time series consisting of hundreds of thousands of observations may be visualized by resorting to pixel-based rendering paradigms [2].

In this paper, we address the problem of generating appropriate visualization layouts for simultaneously viewing large sets of time series using the familiar bar or line charts drawing methods. Our goal is to allow an analyst to quickly perceive relative importance and hierarchy relations within sets of time series, while at the same time supporting good comparability of the data by highly regular layouts.

Our contributions are (1) to introduce the idea of importance-driven layout generation for sets of time series, (2) to formalize a set of constraints that an effective layout for comparative analysis tasks on large time series data should provide, and (3) to provide an efficient algorithm that optimizes the above criteria. This paper is organized as follows: Section 2 introduces the idea behind the importance-driven layout generation for time series data. Section 3 gives a formalization of the problem. Section 4 introduces a family of heuristic algorithms for generating layouts of non-hierarchical as well as hierarchically organized sets of time series. Section 5 presents applications of our system to real-world datasets. Section 6 compares our approach with an aspect ratio aware space-filling layout algorithm, and Section 7 concludes and outlines future work.

2 BASIC IDEA

This section discusses our main objectives for laying out large collections of time series data.

2.1 Concept of Importance Relationships of Time Series

When considering comparative analysis tasks on collections of time series, often there can be perceived a partial or total intrinsic *importance* (or interestingness) relation among the different time series. Such importance relationships should be reflected in the layout. For example, in a sales analysis application, the primary importance measure might be the total sum of sales numbers in each time series. In a network monitoring application, importance relationships may be derived from certain performance metrics taken from hosts on a network. Or, in a stock trading application, importance relationships may be derived from the variance in the stock price time series (a risk measure). An effective layout should support the perception of importance relations by using the two in our opinion most important display properties: *position* and *size*.

Regarding position, usually, the objects at the top of a display are perceived to be more important than those at the bottom, and objects on the left hand side are considered to be more important than those on the right hand side in a given display row (as subject to convention). Regarding size, larger objects are perceived to be more important than smaller objects. These natural ways to reflect importance relationships enable an analyst to quickly locate the most important objects as the data set grows large. In Sections 3 and 4, certain importance measures derived from time series data serve as input for our layout algorithm, which in turn allocates size and position of display partitions into which to place the time series.

[1] {umeshwar.dayal, ming.hao}@HP.com
[2] {keim, schreck}@informatik.uni-konstanz.de

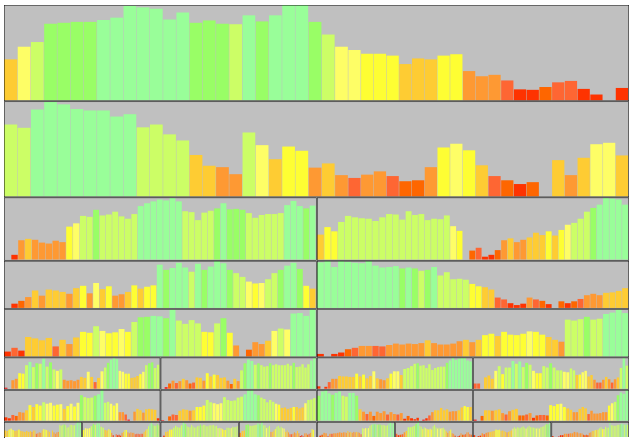

Figure 1: Importance-driven layout of 24 stock price time series with favorable aspect ratios and high overall regularity generated by our algorithm (Section 4). Size and position of the bar chart bounding rectangles approximately indicate the importance relations. The color map and the bar height indicate normalized stock price.

We note that our approach is inspired in part by the *degree of interestingness (DOI)* [8][7] concept. The DOI concept models the interestingness of each data element in a data set as a function of it's *a-priori* interestingness, and it's distance to one or more current *focus* centers. The DOI concept can be used to generate interactive focus-and-context displays using distortion techniques. In terms of the DOI concept, here we only consider the a-priori interestingness component in the data set.

2.2 Space-Filling Layout of Time Series Data

Time series are usually displayed using bar or line charts. Traditionally, multiple time series are accommodated by overlaying them in one common chart, or by using tabular, equal-sized layouts. Both approaches are problematic due to clutter and occlusion (overlaying) and an emerging need for scrolling interaction (tabulating) as the data set grows large. Also, the possibilities for encoding importance and hierarchical relationships are limited in these approaches. We therefore propose an overlap-free space-filling approach to time series layout to addresses both importance-coding and scalability. Overlapping layouts are also possible to address scalability, but we currently do not investigate this line of design.

When laying out sets of bar or line charts in a space-filling display, it is not sufficient to allocate rendering space by assigning position and size according to importance, but *regularity* is a vital criterion for comparing time series. Regularity consists of the aspect ratio, which should be favorable for rendering a given number of time steps within each time series display partition. The aspect ratios of multiple time series should be homogeneous. Also, the alignment of the partitions should be as good as possible, and the number of unique horizontal scales should be low. Experiments we performed suggest that a low number of horizontal scales might be more important than a low number of unique vertical scales. We can support this observation by the fact that in bar and line charts, horizontal scale influences the perception of value sequence and duration of time intervals. Vertical scale influences perception of value magnitudes. While value perception can be easily supported using color maps, supporting perception of time sequence on many different horizontal scales is nontrivial, especially in space-filling layouts.

An additional requirement for importance-driven time series layout arises if there are also hierarchical relationships present among the set of time series. In a sales scenario, for example, the world might be divided into regions, and these regions themselves might be further subdivided into sub regions. For each sub region there may exist a time series for a given product by observing respective sales figures for consecutive points in time. Note that the embedding of hierarchical layout constraints may conflict with the importance-driven layout generation.

Figure 1 illustrates the importance-driven layout of 24 time series from a stock application using our algorithm and assuming a given importance-measure. Our approach achieves a highly regular layout trading off some proportionality between importance and size in favor for the regularity of the layout.

3 Formal Problem Definition

In this section, we formalize certain requirements that an effective importance-driven time series layout should provide. Let $TS = \{TS_1, \ldots, TS_n\}$ denote a set of n time series objects, where a time series TS_i is a set of $|TS_i|$ pairs of real-valued observation with corresponding time stamp. $I_i(TS_i)$ is a real-valued function defined on time series, giving the application-specific, normalized importance measure: $0 \le I_i \le 1 \wedge \sum I_i = 1,\ i \in \{1,\ldots,n\}$. The task of the layout algorithm is to partition an initial (root) rectangular display area R of width $R.w$ and height $R.h$ into a partition $P(R, TS)$ consisting of one sub rectangle $R_i(TS_i)$ for each time series TS_i. Let $|R_i| = R_i.w * R_i.h$ denote the area of R_i, and units are normalized such that $|R| = \sum |R_i| = 1$. Let $R_i.cx$ and $R_i.cy$ denote the x and y coordinates of the center of mass of R_i, with the display origin located in the south-west corner.

3.1 Constraints for an Unstructured Set of Time Series

(1) *Size proportionality constraint.* The area of each time series rectangle should be proportional to the importance of the time series:

$$\sum \big| |R_i| - I_i \big| \rightarrow \min!$$

(2) *Space-filling and non-overlapping constraint*:

$$\bigcup R_i = R \wedge \forall i, j \in \{1..n\}, i \neq j : R_i \cap R_j = \varnothing$$

(3) *Weighted aspect ratio error constraint* as a function of $|TS_i|$ for a user-definable parameter c, which is modeling the relation between time series length and desired aspect ratio:

$$\sum I_i * \left| \frac{R_i.w}{R_i.h} - c * |TS_i| \right| \rightarrow \min!$$

(4) *Ordering constraint.* If $I_i > I_j$, then R_i should either be left of R_j in the same horizontal row, or R_i should be above of R_j. Practically, a threshold parameter ε can be used to decide whether two rectangles are considered to be on the same horizontal row:

$$I_i > I_j \Rightarrow \left(R_i.cx < R_j.cx \wedge \left|R_i.cy - R_j.cy\right| \leq \varepsilon\right) \vee \left(R_i.cy > R_j.cy\right)$$

(5) *Aspect ratio regularity constraint.* Let *unique_ar(P)* be the number of unique aspect ratios in the tessellation *P*. Then:

$$unique\_ar(P) \rightarrow \min!$$

3.2 Additional Constraints for the Structured Case

(6) *Rectangular containment constraint*. Let HU be the set of time series contained in the sub tree rooted at node U in the time series tree (Section 4.3). Let MBR(HU) be the minimum axis-parallel bounding box containing the time series rectangles for HU:

$$\sum_{Treenodes\ U} \left(\left|MBR(H_U)\right| - \left|\bigcup_{i \in H_U} R_i\right|\right) \rightarrow \min!$$

(7) *Hierarchical ordering constraint*. Let HU and HV be two sets of time series contained in two disjoint sub trees rooted at nodes U and V of the time series tree (Section 4.3). Apply the ordering constraint (4) on all pairs of minimum axis-parallel bounding boxes MBR(HU) and MBR(HV).

3.3 Solving the Problem

The above constraints are a postulation regarding certain properties that our layouts should provide. We recognize there exist conflicts between the criteria. E.g., it will not be possible to find a layout that simultaneously realizes the optimum for the error functions in (1) and (5) for most data distributions, given that we obey (2). So, theoretically, we would only be able to find layouts optimal in the Pareto sense, and have to select one from these such that an appropriately combined, scalar error function is optimized. Practically, finding optimal solutions involving multiple competing objectives of this type is a complex problem for which we do not expect to find efficient algorithms. In Section 4, we therefore propose a heuristic algorithm which is motivated by the above constraints, and which is producing visually satisfying results in interactive time. In addition, the defined criteria may serve to experimentally evaluate the effectiveness of layout algorithms with respect to these criteria.

The role of the *importance measure* I_i (for short: *i-measure*) is to impose the importance relation on the set of time series. Usually, appropriate i-measures depend on the specific application context in which the visualization is to be deployed, and will have to be obtained from a domain expert. Suitable I-measures may be as simple as the *min* or *max* aggregation functions (e.g., when monitoring for network performance bottlenecks), or they may involve complex time series analysis algorithms (e.g., when searching for certain local patterns in trading data). In our system, we have implemented a set of basic i-measures, which already serve well for many applications:

- Average, sum, min, max, count, deviation;
- Exception count for some preset threshold;
- Count of local extreme in the time series;
- The average difference between adjacent values.

In addition, a number of optional time series preprocessing methods have been implemented, e.g., offset and amplitude normalization, smoothing, and missing value interpolation.

4 Layout Algorithms

In this Section, we give an efficient recursive importance-driven mapping algorithm (*ID-Map*) by introducing the notion of display masks, and considering unstructured and structured sets of time series data.

4.1 Display Mask Selection and Splitting Policies

Our algorithm recursively maps ordered subsets of time series data into display partitions, which are constructed by a set of so-called *display masks*. A display mask is a scheme for partitioning any given rectangle into a certain number of sub rectangles, reflecting importance-relations by size and position of the sub rectangles as given by the mask definition (mask structure). For allocating a given set of time series, a *mask chooser* first analyzes the distribution of i-measures, and then selects from a set of predefined display masks the mask best accommodating the present distribution of i-measures. We start by defining two masks suited for two salient types of distributions. The *uneven* mask contains three partitions and is appropriate when the distribution of i-measures is skewed. The other mask is the *even* mask and is selected if the distribution of i-measures is rather uniform. We use *Pearson's Mode Skewness* (PMS) [12] as the skewness scale. Figure 2 illustrates.

Considering mask split-point determination, we define three different policies, each one implementing a certain trade-off between size-proportionality and regularity. Policy *A* splits an input rectangle at fixed relative positions, irrespective of the i-measures underlying the data to be allocated. For the even mask, policy A splits both horizontally and vertically at 1/2 edge length. For the uneven mask, it vertically splits at 2/3, and horizontally at 1/2 edge length. Policy A results in maximum regularity, but does not guarantee linear reflection of importance-relationships purely by size. In policy *B*, the rectangle is split vertically in linear proportion to sums of the underlying i-measures, but horizontally it is split at 1/2 edge length. This results in less regularity, but improves size-proportionality. Finally, policy *C* performs all the splitting in linear proportion to the sums of underlying i-measures, guaranteeing linear size-proportionality at the expense of regularity.

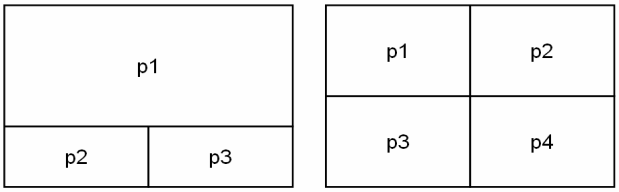

Figure 2: Uneven and even splitting masks (split policy A).

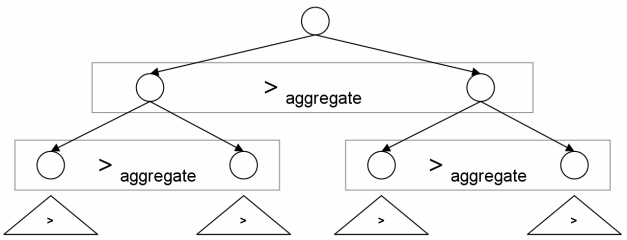

Figure 3. Totally ordered, rooted time series tree.

We note that for now, we fix the splitting policy based on user preference when generating the layout. We note that it would also be possible to perform the policy selection in a data-dependent way, but leave this for future work. Regarding the size proportionality and regularity tradeoff, we note that the importance relations are always encoded in the overall nesting structure of the display, and specialized techniques supporting nesting structure perception exist [16].

4.2 Algorithm for Unstructured Sets of Time Series

For generating an importance-driven layout for an unstructured set of time series, we first determine the i-measure for each time series object, and build a list of time series sorted decreasingly by i-measure. Evaluating Pearson's Mode Skewness of the i-measure distribution of the list, we select the appropriate display mask M_S from a set M of predefined masks. As M_S defines $n=|M_S|$ partitions, we also partition the sorted list of time series into n equal-sized ordered subsets of time series. We then assign each time series subset in order to the respective display partition as defined by M_S, and recursively proceed with all subsets and display partitions, until each time series has been allocated to one display rectangle each.

4.3 Algorithm for Structured Sets of Time Series

A set of hierarchically organized time series can be held in a rooted tree: Inner tree nodes encode the hierarchy; each leaf node of the tree holds one time series. The tree can be totally ordered by i-measures. To this end, we first aggregate the i-measures from all leaf nodes bottom-up along the hierarchy, until each inner tree node is labeled with an aggregated i-measure. We then sort the children of all inner tree nodes by their respective i-measure labels, obtaining a totally ordered rooted time series tree. Figure 3 illustrates.

The algorithm from Section 4.2 can then be applied by considering sorted lists of time series tree nodes, instead of sorted lists of time series. Generation of the layout is initialized by inputting the tree root to the algorithm, and it terminates once all branches of the tree have been processed, and all time series have been allocated. Figure 4 illustrates the allocation of inner tree nodes from a geo-related hierarchy to the partitions of an uneven display mask. The example assumes that the region West has a significantly higher aggregated i-measure than regions North and East. Figure 5 gives the algorithm for the hierarchical case in pseudo code. As will be shown in Sections 5 and 6, this scheme is able to produce regular layouts which favor importance-driven perception and fast visual comparison of many different time series simultaneously.

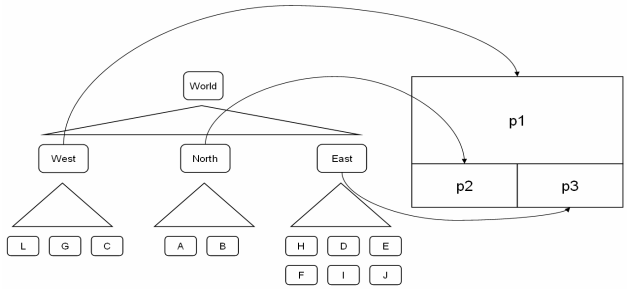

Figure 4: Allocation of inner tree nodes to display space. In this example, the distribution of aggregated i-measures at nodes West, North, and East is assumed to be significantly non-uniform. Therefore, an uneven mask is chosen for the layout of this level in the hierarchy.

5 Application

We have integrated the ID-Map algorithm into a data mining visualization system [4] at Hewlett-Packard Laboratories (HPL). To make large volumes of time series datasets easy to explore and interpret, the system provides many interactive capabilities. The user may set the attributes for hierarchically partitioning the dataset, as well as the color map and time intervals. I-measure and layout parameters can be changed on the fly to analyze data from different perspectives. Drill-down techniques allow to view the data in detail over certain time intervals and sub hierarchies. The system provides interactive update rates and can also accommodate real-time data streams.

We applied the ID-Map technique to a number of real-world sales, network monitoring, and finance datasets. The results

Input:

- Totally ordered rooted tree T with i-measure labels at all tree nodes, and one time series at each leaf node;
- Set M of predefined layout masks, where each mask M_i in M implements an ordered partition of rectangular space R into $n=|M_i|$ sub rectangles $R_1,\ldots, R_n$.

Layout generation:

- The layout is generated by calling `ID_Map(root, rectangle)`, where `root` is the root node of T, and `rectangle` is the initial display rectangle.

```
Global: <Set of display masks> M;

Procedure ID_Map ( <List of nodes> L, <Display rectangle> R ) {
  // terminal node: draw the time series
  If ( L contains exactly one leaf node ) {
         drawTimeSeries( L[0].timeSeries, R );
         return;
  }
  // single non-leaf node: recursively layout child nodes
  If ( L contains exactly one non-leaf node ) {
         ID_map( L[0].children, R );
         return;
  }
  // list of nodes: select display mask; layout chunks of nodes
  Select mask M_S from M such that M_S best represents the
    distribution of i-measure labels from the nodes in L;
  Partition L into n equal-sized, ordered chunks of nodes
    c_1,...,c_n, where n = |M_S|;
  For ( int chunk=1; chunk<=n; chunk++ ) {
         ID_Map( c_chunk, R_chunk( M_S, R ) );
  }
}
```

Figure 5: ID-Map algorithm for the structured case.

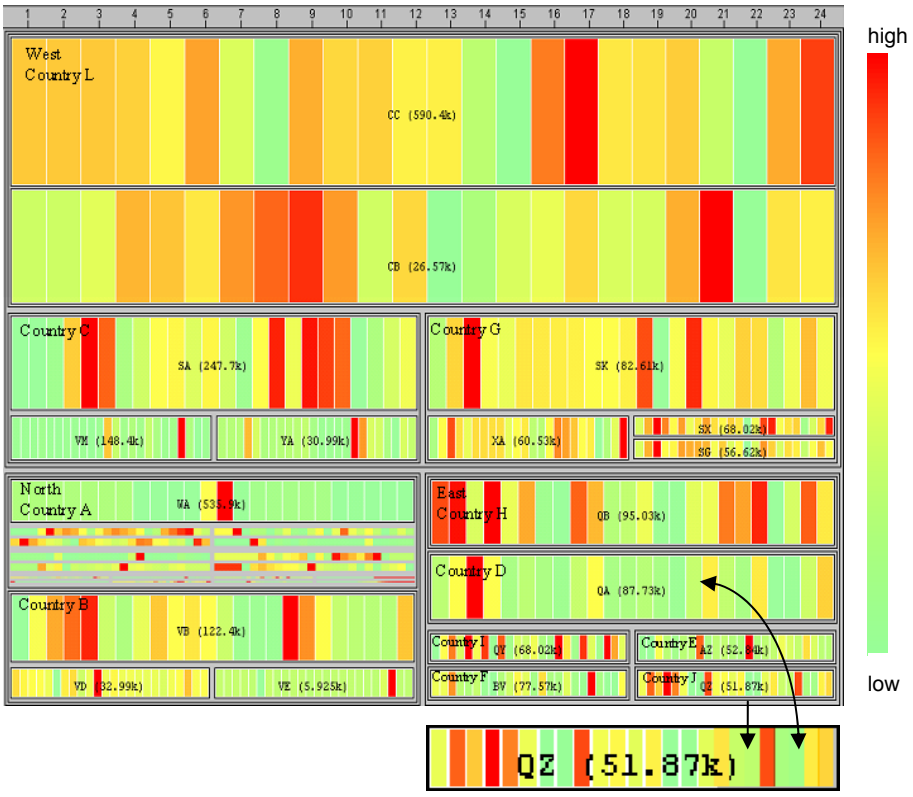

Figure 6: An ID-Map visualizes 35 sales time series and their patterns/trends across three regions: The West region, located at the top of the map, has the most important (highest) sales, more than the North and East regions. We performed a zoom in on state QZ for easy comparison with state QA in the East region. The color map indicates high (medium, low) values of $amount by red (yellow, green)

show the wide applicability and usefulness of this visualization technique. For example, using the ID-Map, the finance team can compare daily sales patterns and trends from each time series for planning sale promotions. The Internet service group can take preventative actions after realizing emerging network traffic bottlenecks. Financial analysts can use technical indicators such as volatility or momentum indicators to layout sets of stock prices from many different industries.

5.1 Sales Analysis

One of the common tasks of sales managers is to improve product sales based on analysis of historic data. Analysts need to know what the dominant daily/monthly/yearly sales patterns are. Which location has continuously high sales, and from which products?

To answer such questions, we can construct ID-Maps like in Figure 6. This map is generated from a real-world sales dataset from last November (41,778 invoices). The dataset has three sales regions (West, North, and East). Each region has a number of countries (e.g., L, C, G in the West region). Each country has a number of states (e.g., CC and CB in Country L). There are a total of 35 time series, placed by the aggregated sales value.

The general comments of the sales teams in our experiments were very positive. They liked to use the system, and especially noted it's applicability in the following use cases:

(1) *Summarization of overall sales time series in one image, ordered by i-measures*. From the size and position of the time series shown in Figure 6, the analysts can instantly find three sales regions: West (top sales), North (medium sales at the bottom left) and East (low sales at the bottom right). Also, they can quickly find that the sales in Country L, state CC are continuously high (more orange and red).

(2) *Multiple time series comparison*. Time series are regularly arranged, with good aspect ratios. Analysts can easily analyze many different time series in one display without overlap. They can zoom a subset of time series to an equal scale for comparison.

(3) *Hierarchical drill down capability*. ID-Map allows the analysts to drill down to the next level of the hierarchy to find the source of the high sales. Figure 7 shows a drill down from Country L, state CC.

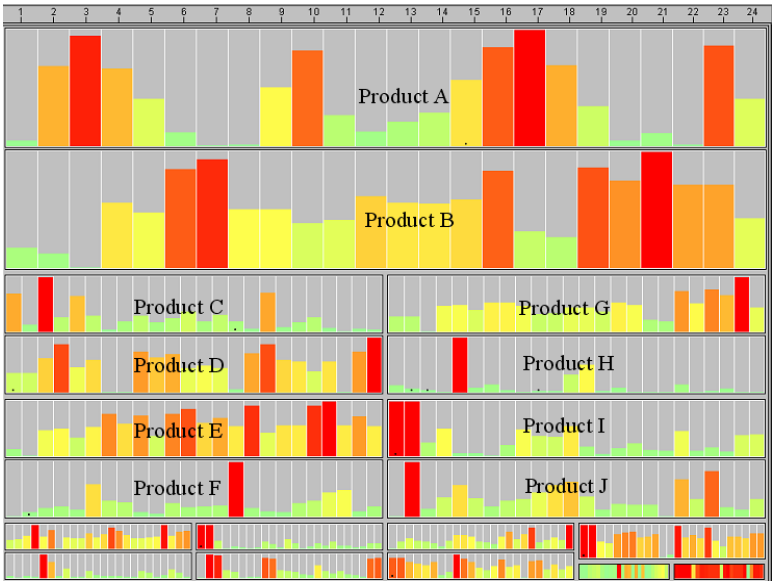

Figure 7. An ID-Map of product sales time series. This map was generated by drilling down Country L, state CC by product type, to observe 19 product sales time series in colored bar charts. Product A (top of map) has the most important sales figures in Country L, state CC.

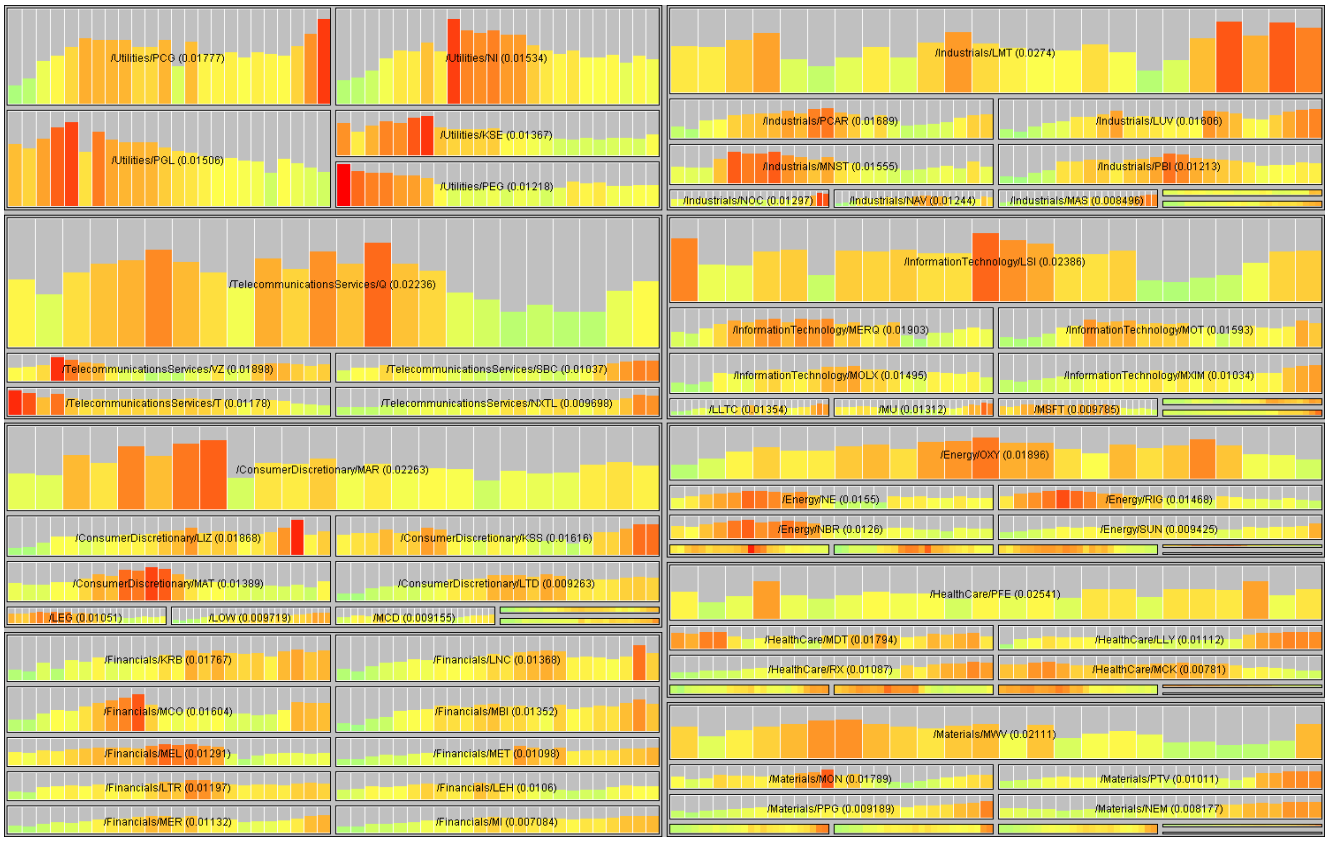

Figure 8: Hierarchical ID-Map (split policy A) of a set of 80 time series from 9 different S&P500 Industries. The mask chooser uses even and uneven masks to distinguish skewed from uniform stock risk among and within Industries. The i-measure used is normalized volatility of stocks; color used in the bar charts indicates normalized stock open price from green (low) through yellow (medium) to red (high).

5.2 Stock Analysis

Finance is an application domain where naturally, large sets of hierarchically organized time series data occur. For example, the GICS standard is a multi-level hierarchical classification of the 500 stocks compound in the Standard&Poor's 500 index. Investors, in order to make investment decisions, often need to overview, compare, and analyze stocks from specific sectors they are interested in. Our algorithm is suited to provide an effective overview over sets of categorized stocks, where the most important time series are readily perceivable and analyzable. Figure 8 shows 30 normalized daily opening prices from October and November 2003 of 80 stocks from 9 different S&P500 Industries (the data was obtained from [10]).

For this example, assume we are interested in comparing the relative volatility of stocks. In order to identify stocks of interest to a risk-seeking investor, we first apply offset- and amplitude normalization [11] as preprocessing. We then select as the i-measure the average difference between all adjacent values of each time series, respectively. This i-measure rates the volatility of stock prices. For sector level aggregation, we average over the i-measures from all time series contained in each of the 9 sectors, respectively. Using this i-measure to guide the layout, the ID-Map algorithm maps each two or three sectors to one partition of an even mask. From the generated layout we learn that the Utilities and Telecommunications Sectors (top-left) are the most volatile in this example, as they are placed into partition *p1* of an even layout. Sectors Energy, Health Care, and Materials (bottom-right) are the least volatile (placed in p4). The stocks from Utilities are more volatile than those from Telecommunications on average (they are placed on top of Telecommunications). On the other hand, the distribution of volatility among the stocks from Utilities is more uniform (ranging *0.018…0.012*) than those from Telecom (ranging *0.022…0.010*). This is readily perceivable, as the mask chooser lays out the Utilities stocks in an even mask, while it chooses an uneven mask for Telecommunication. Using the fixed splitting policy *A* in the layout algorithm, the overall display partitioning is highly regular. Note that only three different x-axis scales are present; this supports visual comparability of the actual time series data.

6 Evaluation

In this section, we compare the ID-Map layout algorithm with an aspect ratio optimizing variant of the well-known TreeMap layout algorithm. We exemplarily discuss key features of both algorithms, and present results obtained from batch experiments performed on synthetic data.

6.1 Aspect Ratio Aware TreeMaps

TreeMap [15] is an excellent tool for displaying large volumes of low-dimensional, hierarchically organized data. By design, the original algorithm does not care about the regularity of the display it generates, as split points are placed in linear

proportion to sums of underlying measures. While proportionality is guaranteed, there is no guarantee for regularity. Several variants optimize aspect ratios [5][6][13]. Here, we extend the stripe-filling method from [6] to support data-dependent target aspect ratios. The original approach aims to generate square rectangles, and so the overall aspect ratio of the rectangle to be filled determines the orientation of the current layout stripe. Here, we consider four possible layout orientations in each step (Figure 9, left). We fill in each orientation stripe with rectangles until the cumulated sum of weighted aspect errors (Section 3.1, constraint 3) is starting to increase. We then make the stripe with the lowest cumulated error permanent, and continue the layout within the remaining rectangle.

We applied this modification to a data set consisting of 24 unstructured time series (48 values each), and assuming a synthetic, square-like distribution of i-measures following $f(x)=x^2$ in [0..1] for each set of time series. For the aspect ratio error function, we set parameter c to 1/16. Figure 9 (right) shows the result. While we have good horizontally-oriented aspect ratios, due to splitting in strict proportionality, the number of different x- and y-axis scales is high. Also, the rectangles are not aligned at stripe borders. Furthermore, the ordering of elements may be discontinuous at stripe borders. These facts result in just medium overall regularity of the display.

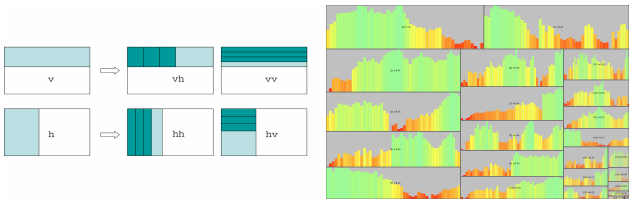

Figure 9: The layout obtained using a modification of the algorithm given in [6] (right). While aspect ratios are good, overall regularity is rather low. The left image illustrates the striping orientations considered during the layout.

6.2 ID-Map Layout

Figure 10 (left) shows the application of ID-Map using the uneven mask with splitting policy *A* (fixed split points) on the dataset from the preceding Section. We notice the display's high regularity. Due to the recursive allocation of chunks of time series to display partitions using fixed splitting, rectangle sizes are not guaranteed to be in linear proportion to i-measures. Even a reversal of size and i-measure relations may occur (note that i-measure relations are always perceivable by the overall splitting structure). To improve proportionality, it is possible to re-sort the rectangles by size in a top-down/left-right manner followed by re-assigning time series to rectangles (Figure 10, right). While we maintain the high regularity of the display, we obtain better proportionality and avoid the reversal case.

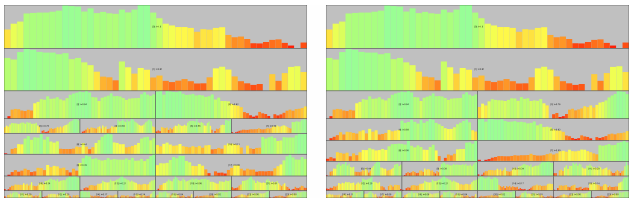

Figure 10: ID-Map using splitting policy *A* (left: standard; right: after resorting of rectangles). Size proportionality is traded off for high overall display regularity.

Splitting policies *B* and *C* increasingly trade-off regularity for improved proportionality between i-measure and rectangle size. The regularity will be reduced, as either vertical (*B*), or both vertical and horizontal splitting (*C*) is performed in proportion to underlying i-measures. To moderate the loss in regularity, we can quantize the split points by implementing a 'snap to grid'-like splitting function. Figure 11 shows the quantized uneven layouts obtained from split policies *B* and *C*. Comparing Figures 10 and 11, we note that policies *B* and *C* improve the i-measure to size proportionality at the expense of reduced regularity, as due to the higher number of different edge lengths occurring, the aspect ratio constraints (constraints 3 and 5 in Section 3.1) get more stressed.

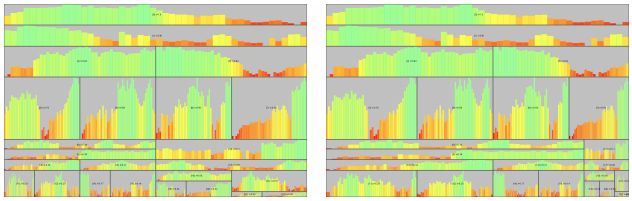

Figure 11: ID-Map using splitting policies *B* (left) and *C* (right). Split points were snapped to the nearest positions on an auxiliary raster grid.

6.3 Experimental Comparison and Summary

We also performed batch experiments to obtain numeric performance results. We used the algorithms to layout unstructured sets of 5 up to 200 time series with 48 values each, within a *1280x929* pixel display, again assuming the square-like i-measure distribution for each set. We restricted ID-Map to use the uneven splitting mask with non-quantized split point determination throughout the experiments (we obtained similar results with the even mask).

Figure 12 shows the normalized ordering error (constraint 4 in Section 3.1) as the fraction of ordering violations among all pairs of time series, using a 10 pixel threshold. Figure 13 shows the normalized aspect ratio regularity error (constraint 5 from Section 3.1) as the fraction of unique aspect ratios among all display rectangles. In both metrics, all ID-Map splitting policies outperform the aspect ratio aware striping algorithm.

We note that in other metrics, the striping algorithm performs quite well. It guarantees size proportionality (constraint 1 from Section 3.1), and does a good job at optimizing data-dependent

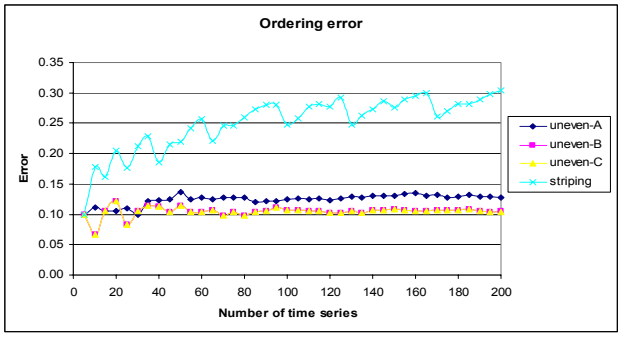

Figure 12: Ordering error results.

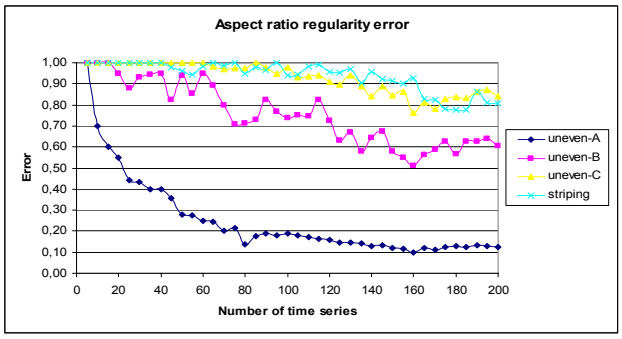

Figure 13: Aspect ratio regularity error results.

aspect ratio targets (constraint 3 from Section 3.1). Also, the number of unique horizontal scales it produces may be low if the algorithm, as depending on the input data, is likely to often choose layout orientations *vv* and *hv* (see Figure 9).

To summarize the comparison, we note that ID-Map provides high display regularity in terms of good rectangle alignment and a low number of different aspect ratios. It largely obeys the top-down, left-right ordering of elements in it's layout. By configuring the algorithm to different splitting policies, the user can control the tradeoff between size to importance proportionality and display regularity according to preference. We point out that in many applications it is well possible to trade off linear proportionality between time series importance and rectangle sizes for regularity. This is because often, appropriate importance measures are just approximate, and sometimes even ordinal, scales for importance relations among time series data, so perfect quantitative reflection might not be meaningful anyway.

7 Conclusion and Future Work

In this paper, we have proposed a novel recursive algorithm to layout large sets of time series data by their relative importance in a space-filling manner. The generated displays possess high regularity and horizontal aspect ratios, making them suitable for time series visualization. We defined a number of objective criteria that suitable solutions should provide. We demonstrated the usability of our approach by applying it on two real-world datasets, and by comparing it with an aspect ratio aware TreeMap variant. Applications are manifold, with prominent examples in the business analysis, finance, and network monitoring domains, among others.

Future work involves improving the system by exploring additional splitting masks, as well as data-dependent methods to quantize split points to simultaneously support regularity, size-proportionality, and ordering objectives. In certain application domains, high data update rates are given which dynamically change underlying i-measures and thus, call for dynamic updates to the display. How to provide good transitions for such necessary structure updates in a monitoring visualization is an open problem which we would like to address. Regarding time series visualization, we plan to extended the system to also support *long* time series which otherwise would not fit lossless on the screen, given the limited display resolutions offered by current displays. Finally, we believe main principles of this approach will extend to other regularity requiring data types as well, e.g., to sequences of images returned by a multimedia similarity search system. We also plan to investigate this direction.

Acknowledgements

Many thanks to Kris Halvorsen of HP Laboratories for his encouragement and suggestions, and to Peter Wright of HP Finance for providing comments and data. We thank Christian Panse, Joern Schneidewind, and Mike Sips from the Information Visualization and Data Mining Group at the University of Konstanz for valuable discussions. We also thank the anonymous reviewers for their helpful comments.

References

[1] J. Abello, H. Schumann, C. Tominski: *Axes-Based Visualizations for Time Series Data*. Proceedings of the IEEE Symposium on Information Visualization, 2003.

[2] M. Ankerst, D. Keim, H.-P. Kriegel: *Recursive Patterns: A Technique for Visualizing Very Large Amounts of Data*. Proceedings of the IEEE Visualization Conference, 1995.

[3] A. Aris, P. Buono, A. Khella, C. Plaisant, B. Shneiderman: *Interactive Pattern Search in Time Series*. Proceedings of the SPIE Conference on Visualization and Data Analysis, 2005.

[4] J. Baker, U. Dayal, B. Deletto, M. Hao, M. Hsu: *A Java-Based Visual Mining Infrastructure and Applications*. Proceedings of the IEEE Symposium on Information Visualization, 1999.

[5] B. Bederson, B. Shneiderman, M. Wattenberg: *Ordered and Quantum Treemaps: Making Effective Use of 2D Space to Display Hierarchies*. ACM Transactions on Graphics, Vol. 21, No. 4, 2002.

[6] M. Bruls, K. Huizing, J. van Wijk: *Squarified Treemaps*. Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization, 2000.

[7] S. Card, R. Rao: *Exploring Large Tables With the Table Lens*. Proceedings of the ACM Conference on Human Factors in Computing Systems, 1995.

[8] G. Furnas: *Generalized Fisheye Views*. Proceedings of the ACM Conference on Human Factors in Computing Systems, 1986.

[9] P. Hanrahan, C. Stolte, D. Tang: *Polaris: A System for Query, Analysis, and Visualization of Multidimensional Relational Databases*. IEEE Transactions on Visualization and Computer Graphics, Vol. 8, No. 1, 2002.

[10] Historical Data for S&P 500 Stocks: *http://kumo.swcp.com/stocks/*.

[11] E. Keogh: *Data Mining in Time Series Databases Tutorial*. Proceedings of the IEEE Int. Conference on Data Mining, 2004.

[12] Mathworld Online Encyclopedia: *http://mathworld.wolfram.com/*.

[13] L. Nigay, F. Vernier: *Modifiable Treemaps Containing Variable-Shaped Units*. Work in Progress Proceedings of the IEEE Visualization Conference, 2000.

[14] E. van Selow, J. van Wijk: *Cluster and Calendar Based Visualization of Time Series Data*. Proceedings of the IEEE Symposium on Information Visualization, 1999.

[15] B. Shneiderman: *Tree Visualization With Tree-Maps: 2-D Space-Filling Approach*. ACM Transactions on Graphics, Vol. 11, No. 1, 1992.

[16] H. van de Wetering, J. van Wijk: *Cushion Treemaps: Visualization of Hierarchical Information*. Proceedings of the IEEE Symposium on Information Visualization, 1999.