Pargnostics: Screen-Space Metrics for Parallel Coordinates

Aritra Dasgupta and Robert Kosara, *Member, IEEE*

Abstract—Interactive visualization requires the translation of data into a screen space of limited resolution. While currently ignored by most visualization models, this translation entails a loss of information and the introduction of a number of artifacts that can be useful, (e.g., aggregation, structures) or distracting (e.g., over-plotting, clutter) for the analysis. This phenomenon is observed in parallel coordinates, where overlapping lines between adjacent axes form distinct patterns, representing the relation between variables they connect. However, even for a small number of dimensions, the challenge is to effectively convey the relationships for all combinations of dimensions. The size of the dataset and a large number of dimensions only add to the complexity of this problem.
To address these issues, we propose *Pargnostics*, parallel coordinates diagnostics, a model based on screen-space metrics that quantify the different visual structures. Pargnostics metrics are calculated for pairs of axes and take into account the resolution of the display as well as potential axis inversions. Metrics include the number of line crossings, crossing angles, convergence, over-plotting, etc. To construct a visualization view, the user can pick from a ranked display showing pairs of coordinate axes and the structures between them, or examine all possible combinations of axes at once in a matrix display. Picking the best axes layout is an NP-complete problem in general, but we provide a way of automatically optimizing the display according to the user's preferences based on our metrics and model.

Index Terms—Parallel coordinates, metrics, display optimization, visualization models.

1 Introduction

In visualization pipeline models, the visual representation is usually regarded as the end product, even though it is obviously a central part of the process of visualization and visual analysis. A good visualization has to provide a clear picture of the relevant structures to the user. Optimizing a visualization cannot only take the data into account, but has to be done differently for different visualization techniques. Structures that are clearly visible in one technique may be hard to find or cause clutter in another.

Despite the increase in quality and resolution of computer displays, visualization still works in a space with a limited number of discrete pixels. This leads to over-plotting, clutter, and other artifacts. While many of these artifacts are considered undesirable and we usually strive to avoid them, they can also point to interesting structures in the data, like clumping points in a scatterplot or a large number of intersections in the center between two axes in parallel coordinates.

While we tend to think of data visualization as an almost perfectly transparent window into the data, we believe that there is a need to study the appearance of the visualization on a limited-resolution screen in order to understand its own properties and how they influence the way they represent the data. Data-based metrics have been proposed to reduce clutter, cluster data and even dimensions, but those only indirectly address what the user sees on the screen. While the perceptual side of visualization is also being studied, little attention is paid to the way the visualization appears on the display. A level of introspection within the visualization would make it possible to calibrate itself based on the actual rendered image on screen.

We propose *Pargnostics* (*Par*allel coordinates dia*gnostics*) as the bridge between the last stages of the visualization pipeline and the user's perceptual system for the specific case of parallel coordinates. By studying the visual structures that the parallel coordinates technique creates on screen, we support the process of exploration in a way that is tailored to the technique. The structures we describe are what we believe to be relevant for the user's perception of the represented data, based on common tasks performed with visualizations. We do not study the perceptual process itself.

- *Aritra Dasgupta is with UNC Charlotte, E-mail: adasgupt@uncc.edu.*
- *Robert Kosara is with UNC Charlotte, E-mail: rkosara@uncc.edu.*

Manuscript received 31 March 2010; accepted 1 August 2010; posted online 24 October 2010; mailed on 16 October 2010.
For information on obtaining reprints of this article, please send email to: tvcg@computer.org.

Once we understand the visual structures on screen, we can use this understanding to optimize the display to support particular tasks. Our optimization arranges the axes of a parallel coordinates display to maximize or minimize certain patterns. This is a significant problem, given that a dataset with d dimensions would require looking through $\left(\frac{d+1}{2}\right)^2$ configurations to see all possible axis pairings (including inversions) [29]. Basing our metrics on axis pairs, we are able to provide a very efficient optimization procedure that greatly reduces the number of configurations that need to be analyzed.

2 Related Work

While there is no complete framework of metrics to describe visualizations, a considerable amount of work has been done in the area.

2.1 The Case for Metrics

Several authors have argued for the need of metrics in information visualization. Tufte [24] was among the first to propose visual quality metrics for static two-dimensional charts, but their applicability to interactive, screen-based visualizations is limited. Miller et al. [19] argued the overall motivation for metrics and their role in predicting the success of visualization tools being developed. A more concrete work and similar to the approach we follow is seen in Bertini et al.'s work on scatterplots [5], which employs a non uniform sampling technique to reduce clutter in 2D scatterplots. They further argue for the need of metrics [6] where they define different visual quality metrics and provide a research direction for each. The definition of the metrics is applicable to our work, because we conceive Pargnostics metrics as a diagnostic model for visually effective quality and feature-preserving visual structures.

2.2 General Screen-Space Metrics

A key aspect of Pargnostics is that it is an information-assisted visualization model [7], but based on image-space metrics rather than information abstracted from data. Our goal is to convey the different visual structures that encode information to the user. One of the early instances of such work is *Scagnostics* (*Sca*tterplot Dia*gnostics*), proposed by Tukey et al. [25]. This work was extended by Wilkinson et al. with more detailed graph-theoretic measures [30] for detecting a variety of structural anomalies in a geometric graph representation of the scatterplot data. The resulting rating can be used to pick views that show particular structures that are of interest to the user.

Similarly, in Pixnostics [21] the authors use image- and data-analysis techniques in conjunction to rank the different lower-

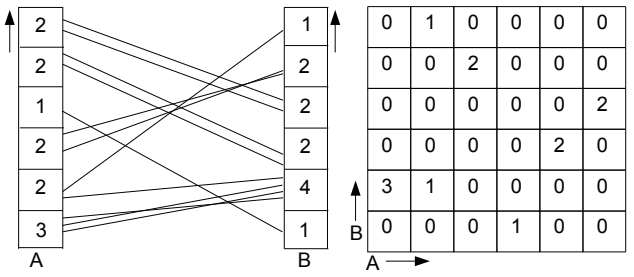


Fig. 1. Illustrating binning with equal-width histograms: On the left is the binned space of the parallel coordinates and on the right is the degree of each bin in the 2D histogram.

dimensional views of the dataset and present only the best to the user. The method creates lower-dimensional projections that provide maximum insight into the data and optimizes the parameter space for pixel-oriented visualizations.

In Pargnostics, we focus on parallel coordinates and, similar to Scagnostics and Pixnostics, aim to reduce the analyst's burden of searching for the optimum views of the data. The over-arching goal of Pargnostics is to play the role of a multi-dimensional detective [12], where we diagnose structures of interest on behalf of the user. In addition, the user is aided with an interactive interface to enhance the effect of helpful structures (e.g., convergence/divergence, parallelism) and reduce the same for structures that hinder his analysis process (e.g., large number of crossings, over-plotting).

2.3 Metrics for Parallel Coordinates

There are some instances of image-space based metrics in the context of parallel coordinates [14, 13]. Tatu et al. propose similarity-based functions based on Hough Space transforms to find clusters [23]. They propose dimension reordering based on analysis functions on the resulting image of parallel coordinates. Johansson et al. [15] propose a screen-space metric based on distance transforms to estimate the visual quality of the abstracted dataset. Our approach is different in the sense that we attach semantics to the structures that can be seen on the screen and not only define the metrics qualitatively but also quantitatively. Dimension reordering has also been addressed in earlier work: Ankerst et al. [4] compute the degree of similarity among dimensions based on data-space metrics and cluster similar dimensions together. A somewhat similar idea is pursued by Yang et al. [31]: similar dimensions are clustered and used to create a lower-dimensional projection of the data. In Pargnostics we focus on dimension reordering as a tool for optimization based on a new set of screen-space metrics. We also employ optimization where the user can select views that suit his analysis by browsing through a rank-ordered view by features.

2.4 Visual Artifacts in Parallel Coordinates

An important building block of Pargnostics is that we investigate how the different visual artifacts in parallel coordinates like the ones described in Section 3 either help or hinder the knowledge discovery process. Crossing of lines is one such important artifact: too few crossings imply lack of correlation and too many can cause clutter. Zhou et al. [32] propose using curved edges instead of straight lines for connections among points between axes, which increases the traceability of lines and thus reduces edge clutter. Ellis and Dix [9] use line crossings and crossing angles to reduce clutter and the number of lines within a user-controlled lens. We do not measure clutter explicitly, but compute the causal factors like line crossings and angles of crossing. Our approach also considers more uses of these metrics than just determining occlusion and clutter.

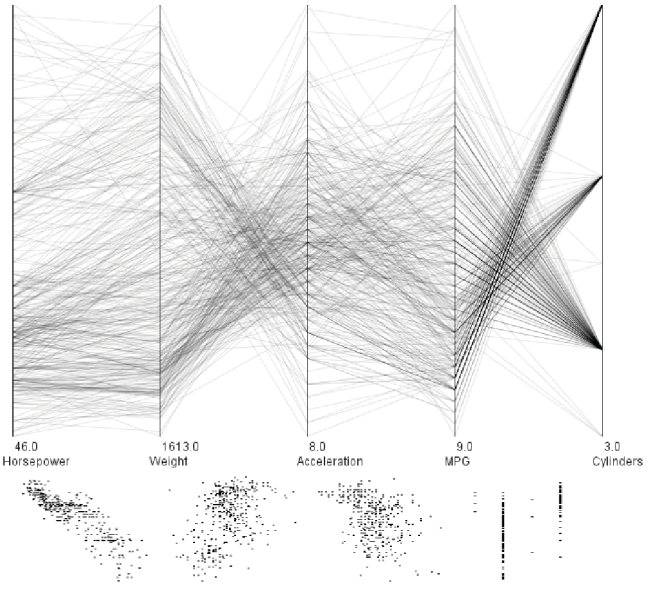


Fig. 2. Scatterplot features mapped to parallel coordinates: The right axis of each axis-pair corresponds to the X-axis of the scatterplot and is labelled accordingly.

3 Metrics

Pargnostics metrics are intended to be the bridge between the visual representation and the analytical tasks of the user. The choice of metrics depends upon the analytic tasks that a user would perform with the tool. Finding correlation and clusters in the data are typically considered to be among the ten most important low-level analytical tasks in information visualization [2]. Parallel coordinates, in particular, supports tasks like examining properties and relationships between variables represented by axes [3]. We designed our metrics with the goal of establishing the missing link between the geometrical features and these analysis tasks: line crossings, parallelism, mutual information (correlation); angles of crossing, convergence-divergence (clusters); over-plotting, and pixel-based entropy (image quality).

Since we are working in a limited screen-space, all the metrics have to be optimized to operate efficiently in real-time. Therefore we provide an optimization framework based on user-preferred settings. We are also cognizant of the visual information seeking mantra [22]. We provide multiple views of the data: the ranked view shown in Figure 10 and the matrix view give an overview of the structures in the parallel coordinates visualization. This is especially useful when the user is totally unfamiliar with the data, or wants to take a fresh look at known data. The main view can then be used to filter and optimize the data, as well as getting details on demand.

Among the metrics, we find the number of line crossings and parallelism metrics to be the most useful for optimization, because they help reduce clutter and maximize the visibility of correlations. Angles of crossing are also helpful for avoiding highly-correlated combinations where line crossings get difficult to read. The convergence-divergence metric helps isolate categorical or quasi-categorical axes (numerical but with only few values); the pixel-based entropy allows us to optimize the alpha value when rendering larger datasets that lead to clutter.

3.1 Model

Our metrics are based on a particular view of parallel coordinates that differs from that of most existing work. This model informs the design of the metrics and also has consequences for the optimization procedure described later in this paper.

Pargnostics are a set of screen-space metrics for parallel coordinates. As such, they inherently depend on the size of the display, measured as the number of pixels. This discussion is based on the common

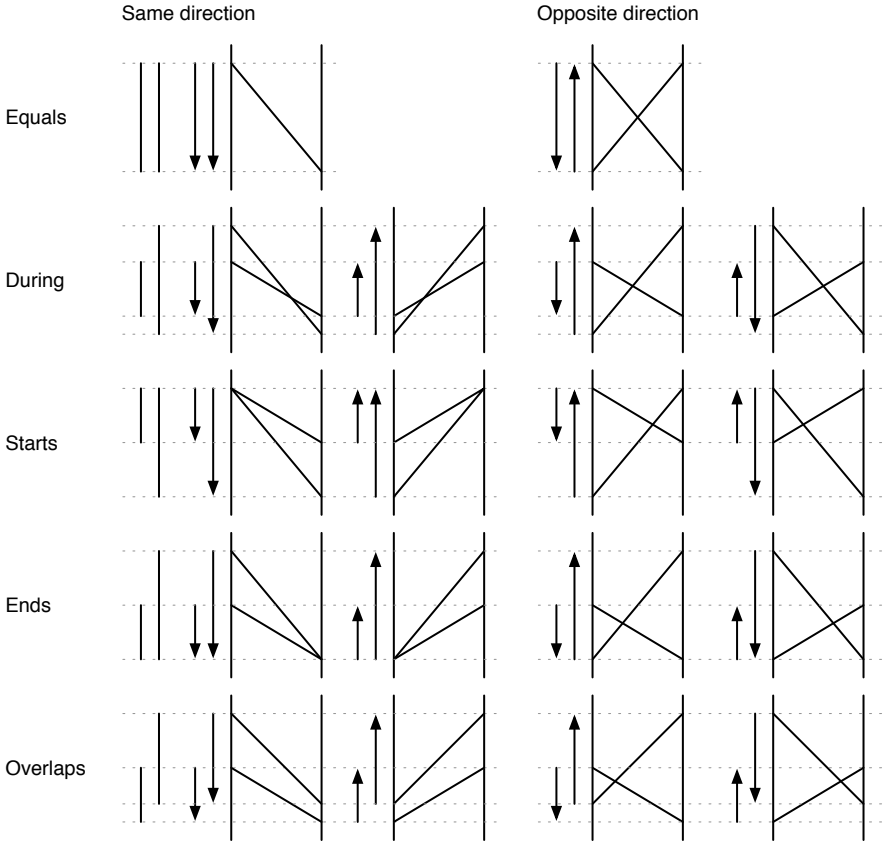


Fig. 3. The possible configurations of two lines in parallel coordinates, classified using Allen's interval algebra. *Before*/*after* and *meets* relations are not shown, because they trivially mean no intersection. See Table 1 for a classifications of crossings using these criteria.

parallel coordinates layout, which draws vertical axes and lays the axes out horizontally from left to right.

We consider a parallel coordinates display to consist of a series of axis pairs, similar to scatterplots in a scatterplot matrix (Figure 2). The space between a pair of axes is where interesting patterns such as parallel or crossing lines, aggregations of lines, etc., can be observed. As pointed out by Li et al. [17] finding correlation in parallel coordinates ultimately boils down to finding those patterns between pairs of axes. This is also true for all relevant structures that the user is looking for in the data. Neighboring axis pairs of course depend on each other because the left or right axis has to correspond to the right or left axis of the neighboring pair, respectively.

Axis inversions are considered within each axis pair, and are independent of each other. Our model is only concerned with the relative direction of axes within the pair, not with their absolute direction as they appear on screen. This enables us to considerably simplify and speed up the optimization process (Section 4). Local axis pair inversions are propagated through the list of dimensions when the result of the optimization is to be shown on screen.

3.2 Pixel-Space Histograms

The basis for most metrics is pixel-based binning (Figure 1). With each axis being h pixels high, we define three types of histograms: one-dimensional axis histograms, one-dimensional distance histograms, and two-dimensional axis pair histograms.

All binning is done in screen space, after the values have been transformed into pixel coordinates. The data values, v_i, are projected onto screen pixel coordinates, l_i (for the left axis in a pair) or r_i (for the right axis), by a mapping function f: $(l_i, r_i) = f(v_i)$. We disregard any padding around the axes, so $0 \leq l_i < h$.

One-Dimensional Axis Histogram. The basic one-dimensional axis histogram has h bins, and consists of bins b_i that count the number of lines starting or ending in them. In the following formula, i is in the range $[0; h-1]$.

$$b_i = |\{k \mid \lfloor l_k \rfloor = i\}|$$

One-Dimensional Distance Histogram The one-dimensional distance histogram records the slope of the lines between the axes, measured as the vertical distance in pixels. This measure can be used to calculate the actual angle when the spacing between the axes is given. The steepest line going up or down covers the entire height of the display, and can thus have a vertical distance in the range $(-h; h)$. This leads to a total of $2h-1$ bins, as the up and down ranges overlap at the value 0. This histogram is used for calculating the parallelism metric.

$$d_i = |\{k \mid \lfloor r_k - l_k \rfloor = i\}|$$

Two-Dimensional Axis Pair Histogram. Looking further at the space between the axes, we define a histogram of all the lines, covering both axes. This is a two-dimensional histogram, consisting of bins b_{ij}, with both i and j in the range $[0; h-1]$.

$$b_{ij} = |\{k \mid \lfloor l_k \rfloor = i \wedge \lfloor r_k \rfloor = j\}|$$

This is the basis for the calculation of the number of line crossings (Section 3.3), crossing angles (Section 3.4), convergence/divergence (Section 3.7), as well as over-plotting (Section 3.8).

3.3 Number of Line Crossings

A prominent feature of parallel coordinates are crossing lines between pairs of axes. Many crossing lines typically mean some kind of inverse relationship, especially if the crossings occur mostly around the

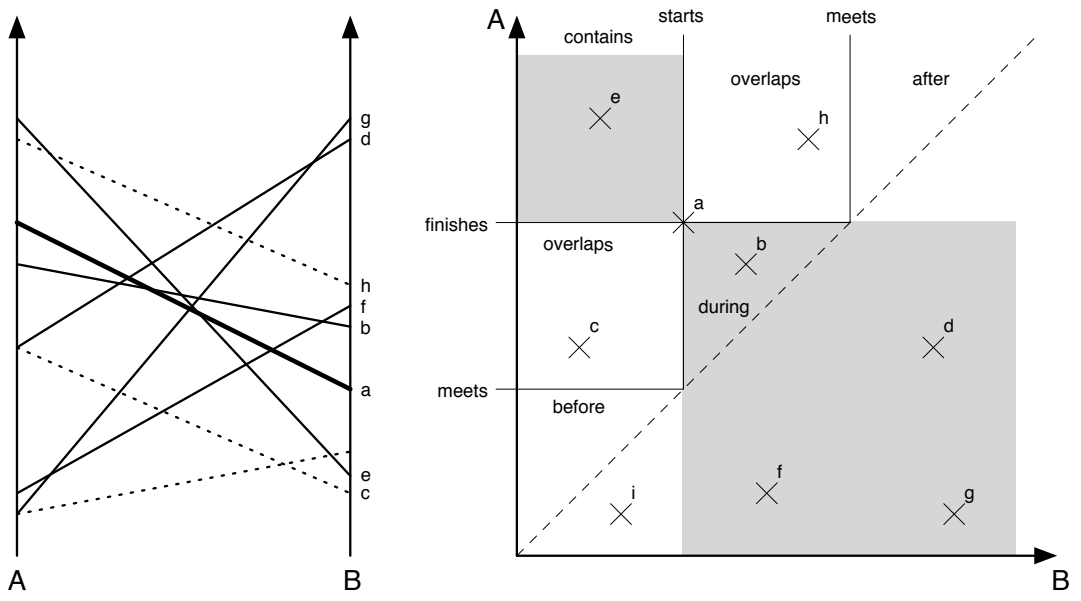


Fig. 4. Using the interval classification, we can simplify the structure by using Rit's sets of possible occurrences [20]. The different lines in parallel coordinates are shown in the scatterplot/SOPO diagram on the right. The thick reference line *a* is used as the reference on the right. All intervals that fall into the shaded areas represent lines that cross the reference line, those lines are drawn as solid lines; dotted lines do not cross the reference line.

center. Line crossings also contribute to clutter and can make it difficult to identify which lines are going where. The importance of line properties in implying correlations have been studied [18, 29]; Ellis and Dix [9] also dealt with line intersections as a cause for display clutter. Also, user studies [17] show that judging correlations can be a complicated task because of several artifacts in parallel coordinates. Since line crossings directly affect correlations, especially inverse correlations, it is important the user is able to maximize or minimize it for optimization.

To efficiently calculate the number of crossings, we interpret each line between a pair of axes as a directed interval. Allen's interval algebra for temporal reasoning [1] enumerates all possible relationships between pairs of time intervals: A *before* B, A *meets* B, A *overlaps* B, A *starts* B, A *finishes* B, and A *equals* B. All relationships except *equals* can also be applied as B *R* A, leading to a total of 13 different ones.

For the purpose of determining line crossings, we add the direction of the line or interval as a second attribute (Figure 3). We are not interested in the absolute direction, but only whether the two lines are pointing in the same or opposite directions. Disregarding the *before*/*after* and *meets* relations, which trivially cannot lead to intersections, we can classify which relationships and directions lead to intersecting lines and which do not (Table 1).

This classification can be simplified by making use of work by Rit, who devised a way to graphically propagate temporal constraints [20]. Similar to the point-line duality between scatterplots and parallel coordinates, sets of possible occurrences (SOPOs) depict intervals as points on a two-dimensional diagram with two time axes: one for the start and one for the end of the interval. Combining all the areas that correspond to the line crossing conditions, we find a simple rule for determining whether two lines cross (Figure 4).

Given the two-dimensional histogram, we can calculate the number of line crossings, L:

$$L = \sum_{i=0}^{h-1} \sum_{j=0}^{h-1} \sum_{k=i+1}^{h-1} \sum_{l=j+1}^{h-1} b_{ij} b_{kl}$$

This leads to a complexity of $O(h^4)$, though in practice it is much lower: if b_{ij} is zero, the program does not need to enter the two innermost loops. In real-world data sets, the histogram is very sparse, so this condition has a large effect on algorithm runtime.

An alternative way to calculate L is by comparing lines directly, using the condition derived from the discussion above.

$$L = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \begin{cases} 1 & r_i < l_i \wedge r_j < l_j \vee r_i > l_i \wedge r_j > l_j \\ 0 & otherwise \end{cases}$$

The complexity of the latter is $O(n^2)$, though since usually $h << n$, the two methods perform equally well for many real-world data sets.

L is bounded by the number of possible line crossings in a given data set of size n. Since each line can intersect with every other line, but only once, the upper bound is $\frac{n(n-1)}{2}$. The normalized number of crossings, L_{norm}, therefore is:

$$L_{\text{norm}} = \frac{2L}{n(n-1)}$$

3.4 Angles of Crossing

While the number of line crossings can be an issue, an equally important one involves the angles at which lines cross. Lines crossing at flat angles are harder to follow than ones crossing at close to right angles [11, 28]. Lines that are part of clusters also tend to cross at low angles [29].

We calculate the crossing angles between pairs of lines. Except for parallel lines, any two lines will have two crossing angles that add up to 180°. Since we care about how close to a right angle the lines cross, we choose the smaller one of the two. The angle calculation is only performed for lines that are known to cross based on the crossing lines conditions above. They are then rounded and binned into whole degree bins, and we calculate the median crossing angle. The resulting histogram (Figure 5) is used for display purposes, while the median is used as the criterion for optimization.

The calculation is based on the two-dimensional axis histogram. For each pair of bins that are found to contain crossing lines, we calculate the crossing angle. This means that for multiple lines crossing

Table 1. Conditions for intersection based on interval relation and direction, grouped and ignoring symmetrical cases (see Figures 3 and 4).

Relation	Direction	Intersection
before/after	same	no
	opposite	no
meets	same	no
	opposite	no
overlaps	same	no
	opposite	**yes**
during	same	**yes**
	opposite	**yes**
starts/finishes	same	no
	opposite	**yes**
equals	same	no
	opposite	**yes**

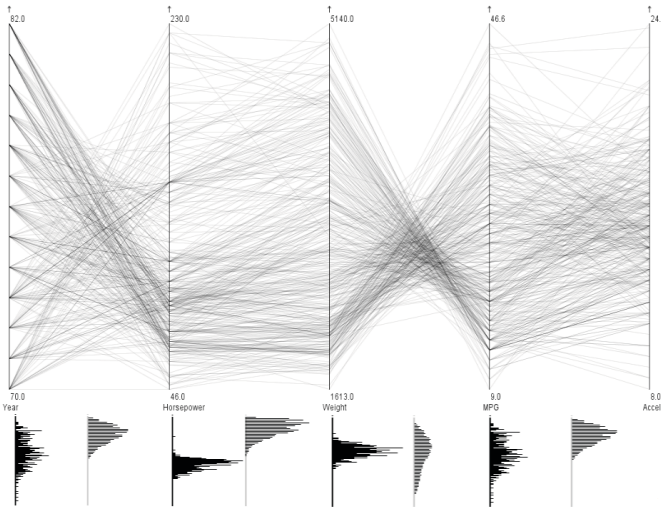


Fig. 5. Distance histograms (left half of each cell below the parallel coordinates) and angles of crossings (right half) histograms for different dimensions of the cars data.

at the same point, we only have to perform the calculation once, and can add $b_{ij}b_{kl}$ to the histogram (similar to the way it is done for the number of crossings above).

3.5 Parallelism

In addition to line crossings, there are also structures where lines are parallel or close to parallel to each other. Such lines can mean correlations between two dimensions [3]. Li et al. described the difficulties user have judging correlations in a parallel coordinates environment [17]. Parallelism can also imply closely aggregated lines or clusters within a subset of the data. When given the choice between line crossings and parallel lines, the latter are often preferable to crossings because they lead to less clutter. This is of importance for the display optimization described below.

To describe parallelism, we compute a vertical distance histogram between any two connecting points on adjacent axes (Figure 5). Positive values in this histogram mean lines going up, negative values indicate lines pointing down. Axis pairs with a high degree of parallelism tend to have narrow distributions of directions, while ones with no or very little apparent parallelism cover the entire distance spectrum with no apparent clustering of values.

Parallelism is defined both in terms of direction and extent. The median distance value indicates the direction and the extent of parallelism is given by the interquartile range: a narrow interquartile range implies high parallelism. We normalize the distances between 0 and 1, by dividing by the highest possible distance. We then compute parallelism P_{norm} as follows based on the interquartile range between the 25% and the 75% quartiles, q_{25} and q_{75} .

$$P_{\text{norm}} = 1 - |q_{75} - q_{25}|$$

The subtraction is done to get a higher parallelism value for a higher degree of parallelism (and thus a smaller interquartile range). The direction is given by the median M_P, which is not normalized (the direction only makes sense in pixel coordinates):

$$M_P = q_{50}$$

3.6 Mutual Information

In exploratory data analysis, the information the user is looking for is subjective: task oriented and difficult to model. Mutual information [8] provides a general measure of dependency between variables; its value is zero when they are conditionally independent. Pearson correlation coefficient is extensively used for this purpose, but a lack of such correlation does not imply that two variables are independent. In Pargnostics, we treat the data dimensions as random variables, and our computation is based on the binned space. The probability that a dimension assumes a particular data value is therefore equivalent to the binned value in the screen space.

Let X and and Y be random variables. Mutual information $I(X;Y)$ is defined in terms of probability distributions as:

$$I(X;Y) = \sum_{i=1}^{h}\sum_{j=1}^{h} p(x_i,y_j)\log\frac{p(x_i,y_j)}{p(x_i)p(y_j)}$$

In this case $p_i = \frac{b_i}{h}$, using the one-dimensional axis histogram, where $p(x_i,y_j)$ denotes the joint probability of random variables X and Y and in this case $p(x_i,y_j) = \frac{b_{ij}}{h}$ using the two-dimensional axis histogram (Section 3.2).

Some of the dimensions which exhibit high mutual information (Figure 10) in the cars data are *MPG* and *weight*, *weight* and *acceleration*, *horsepower* and *weight*, etc. This is expected as we know lighter cars have good fuel economy and better acceleration. Thus mutual information provides an indication to the user, which axis pairs are likely to convey interesting information.

3.7 Convergence, Divergence

A common pattern in parallel coordinates is comprised of few values on one axis that branch out to many values on the other. Similar to *clumpiness* in Scagnostics there can be many lines converging to a point or diverging from a point between adjacent axes. They represent sine functions with multiple periods [10]. These structures are a useful characterization of the data points as they reveal associative relationships: many-to-one or one-to-many relationships between points on adjacent axes.

Convergence. Convergent structures between a pair of axes comprise of those lines on the left axis which converge to a single bin on the right axis. Mathematically, the total convergence C between two axes can be calculated as:

$$C = \sum_{i=1}^{h}\sum_{j=1}^{h}\begin{cases} 1 & \text{if } b_{ji} > 0 \\ 0 & \text{otherwise} \end{cases}$$

Divergence is the mirror image of convergence and is given by

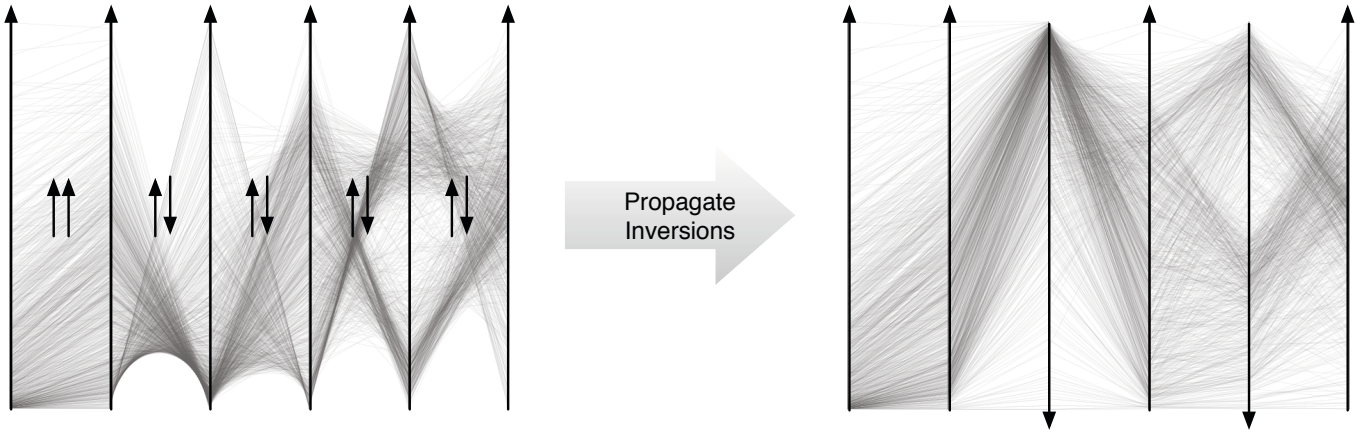


Fig. 6. Axis inversions are handled locally during the optimization, and only considered per axis pair. They are later propagated through all dimensions from left to right to determine which axes end up being inverted.

$$D = \sum_{i=1}^{h} \sum_{j=1}^{h} \begin{cases} 1 & \text{if } b_{ij} > 0 \\ 0 & \text{otherwise} \end{cases}$$

The ratios $C_{avg} = \frac{C}{n}$ and $D_{avg} = \frac{D}{n}$ where n is the total number of lines between two axes, indicate the average degree of convergence or divergence between each axis. To give an overview of the degree of convergence/divergence present between each pair of dimensions, we show histogram of convergence or divergence, whichever is greater. The values are normalized as follows:

$$C_{norm} = \frac{C}{\max(b_{ij})}$$

$$D_{norm} = \frac{D}{\max(b_{ij})}$$

3.8 Over-plotting

Over-plotting is a measure of the quality of a visualization. This is especially relevant in the case of parallel coordinates. Here, due to the large dataset sizes and a limited number of screen pixels, several data points can be mapped to the same pixel value on two adjacent dimensions. The degree of over-plotting helps to estimate the information density between adjacent dimensions. High over-plotting leads to dense cloud of lines and the user might often be interested in a view that minimizes it. Quantitatively, over-plotting is a side effect of binning and is therefore an indication of the information loss that occurs during processing. Having different bin size would directly affect over plotting and help us achieve controlled information loss. The degree of over-plotting O is computed from the count of each bin of a two-dimensional histogram. Each count in a bin greater than 1 contributes to over-plotting.

$$O = \sum_{i=1}^{h} \sum_{j=1}^{h} \begin{cases} b_{ij} & \text{if } b_{ij} > 1 \\ 0 & \text{otherwise} \end{cases}$$

The total degree is then normalized by the number of data points n to give the normalized degree of over-plotting:

$$O_{norm} = \frac{2O}{n(n-1)}$$

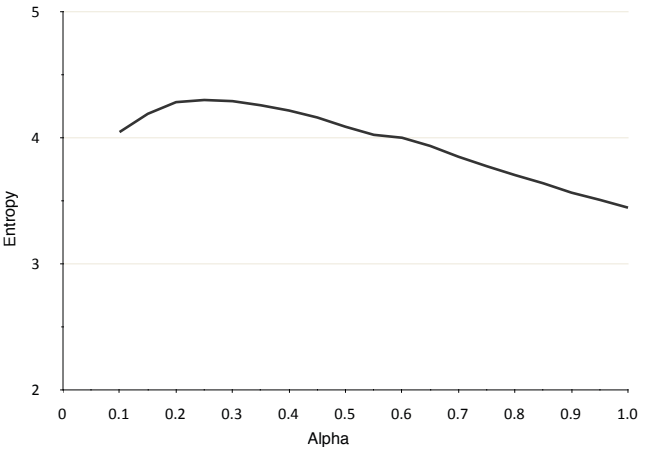


Fig. 7. Entropy for the wine dataset, measured for different α values from 0.1 to 1.0, in steps of 0.05. The maximum at 0.25 yields the most visual information; less is too faint, more leads to clutter.

3.9 Pixel-based Entropy

Entropy measures the degree of randomness in a process, and thus the amount of uncertainty present at any segment of a visualization display. In case of parallel coordinates, we compute the entropy for the region between a pair of axes: regions of high entropy imply high information density [21, 27]. Entropy is computed from a rendered parallel coordinates image, using the gray levels as the alphabet that is being transmitted. We first aggregate the values into a histogram with 256 bins, x_i. The probability of encountering a particular gray value, given the total number of pixels n_{pixels} between the axes, is

$$p_i = \frac{x_i}{n_{\text{pixels}}}$$

Using this definition, we calculate the entropy based on the usual definition:

$$H = -\sum_{i=0}^{255} p_i \log p_i$$

For greater numerical stability (because the p_i tend to be very small), we use $\frac{n_{\text{pixels}}}{x_i}$ in the actual calculation, which modifies the formula into

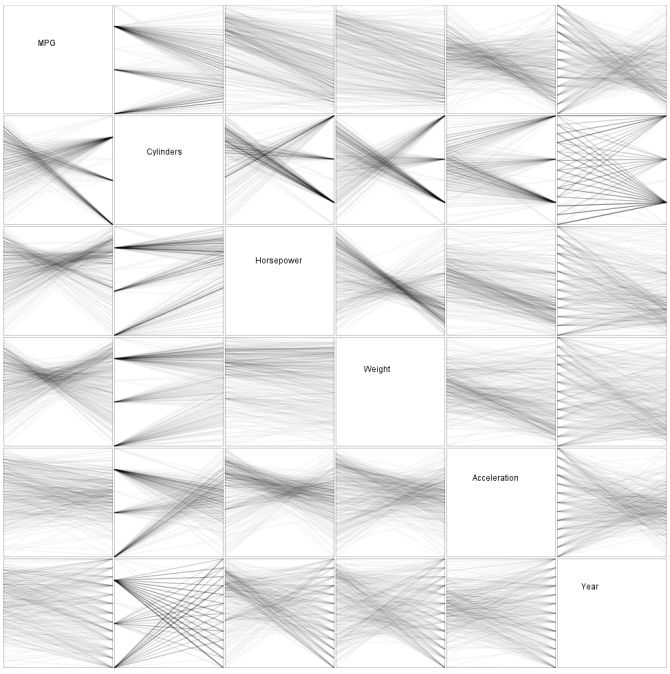


Fig. 8. Parallel coordinates matrix of the cars dataset, showing axes pointing in the same directions in the lower left, and inverted axis in the upper right half.

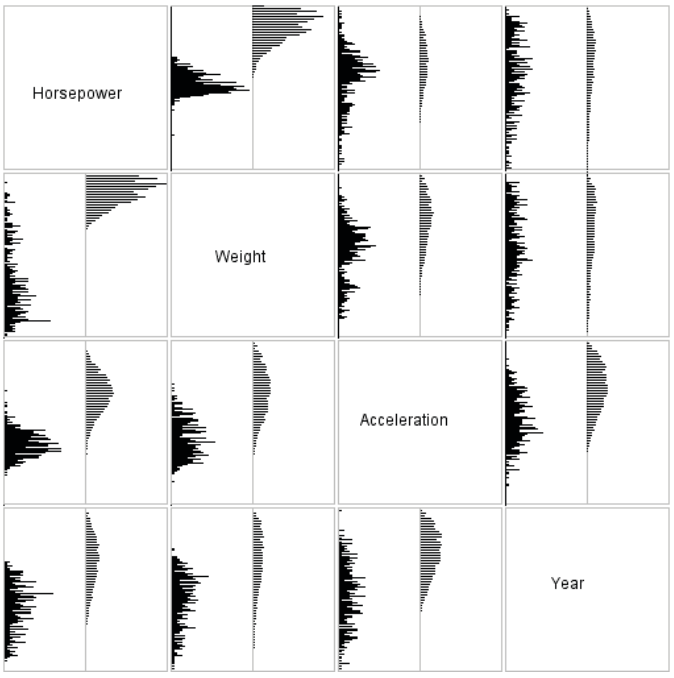


Fig. 9. Histogram matrix for the cars dataset. The left histogram in each cell shows the distance, the right one line crossing angles.

$$H = \sum_{i=0}^{255} p_i \log \frac{1}{p_i} = \sum_{i=0}^{255} \frac{x_i}{n_{\text{pixels}}} \log \frac{n_{\text{pixels}}}{x_i}$$

In parallel coordinates, a high entropy indicates a region with a large number of line crossings, but no inverse structure – an inverse relationship leads to a large amount of white pixels at the top and bottom of the region, which greatly reduces the entropy. While there is no simple connection between entropy and display structures, maximizing entropy generally yields busy but very readable displays of the data. Entropy computed across the entire display also allows us to use semi-transparent rendering similar to density plots for very dense data, and to optimize the α value used (Figure 7).

4 Dimension Order Optimization

Using the above metrics, we are able to optimize the display for the analyst. In general, finding an optimal ordering of axes for parallel coordinates is equivalent to the traveling salesman problem, and thus NP-complete [16]. Using a branch-and-bound algorithm and considering the special properties of parallel coordinates, we are able to find optimal solutions in much less time in general. Our binned data model of parallel coordinates also reduces many the computations (reducing the impact of the number of data items) and even handles axis inversions as local decisions, leading to a very efficient solution.

We use a branch-and-bound algorithm to find the optimal order of axes. We first build a matrix of all axis pairs and the cost associated with them. The cost can be a combination of several metrics, using weights selected by the user. This computation is only performed once, and is the only step that depends on the number of records in the dataset. All subsequent steps are performed on the basis of this matrix and thus only depend on the number of dimensions.

4.1 Axis Inversions

Axis inversions are handled per axis pair as part of building the cost matrix. For each axis pair, the cost for both the inverted and the non-inverted situation are computed across all the desired metrics. The lower value is used in the matrix, and the program records which of the two values that was.

Axis inversions can be handled locally because we consider them only between the two axes in each axis pair. Inverting one axis pair does not have an immediate effect on neighboring axis pairs. Only once the optimal solution has been found, the inversions are propagated across the axes from left to right (Figure 6). Since users more likely prefer axes pointing up, the program also performs a global inversion that flips all axes if the majority of them are pointing down after the propagation.

While axis inversions can dramatically reduce the number of crossings and increase parallelism, they also make reading the visualization more difficult, because they require the analyst to look up the axis orientations and remember which ones are inverted and which are not. The optimization therefore provides the option not to use inversions in the process.

4.2 Branch-and-Bound Optimization

We implemented the optimization as a branch-and-bound algorithm that uses a priority queue and best-first search. A key issue in branch-and-bound implementations is how tight the bounds can be estimated when the decision is made about whether to cull a sub-tree or not. In our case, these estimates are based on a full cost matrix, and are very precise.

Based on our case studies, we find that the best-first search very quickly finds the optimal solution after only a few complete solutions (i.e., having walked through to a leaf of the search tree). This is not surprising given the reduction of the metrics to axis pairs, rather than looking at the entire visualization at once. In most cases, the optimization ends up only picking the best next axis from the ones still available, rather than having to evaluate a much larger number of permutations.

5 Performance

For practical use, metrics need to be calculated in reasonable time. The fact that all Pargnostics metrics are based on histograms simplifies computation, and reduces the real-world complexity by reusing intermediary results.

All histograms are created in one pass, with complexity of $O(n)$ (n being the size of the dataset). In our implementation, the one- and two-dimensional histograms are computed at the same time, which further

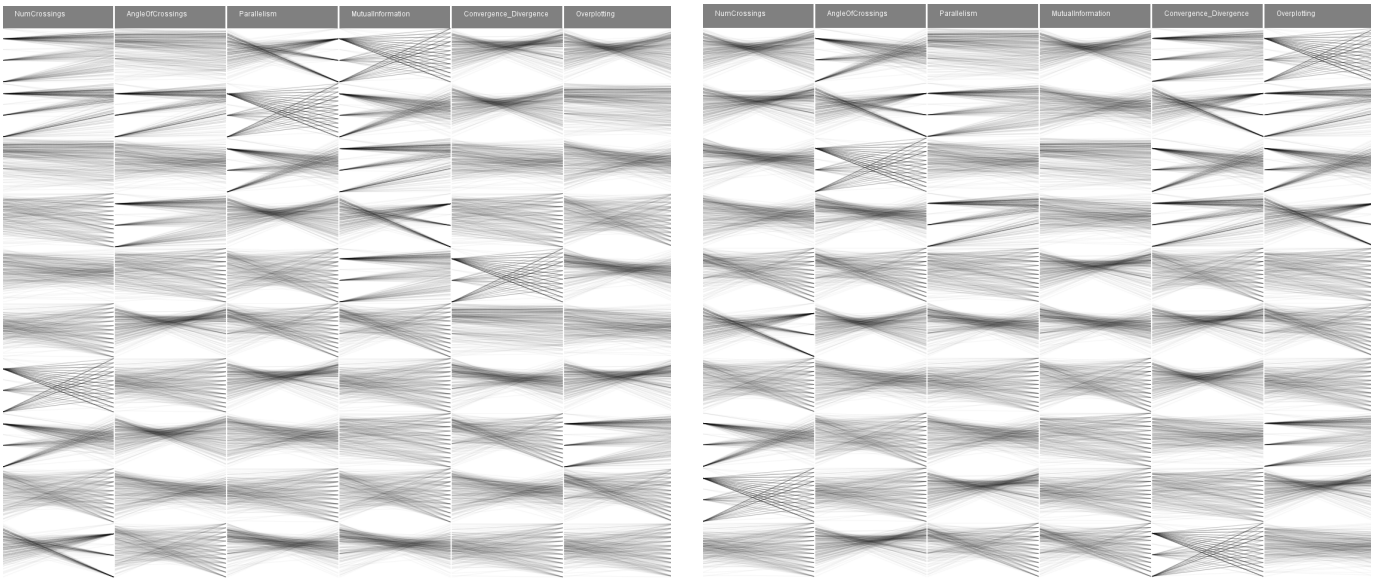


Fig. 10. Parallel coordinate views of the cars dataset, ranked by different metrics. The left view shows ascending order, the right one descending order. In our implementation, the user is shown the names of the axis when mousing over a plot, and the display also highlights all instances of the same plot the user is pointing at.

reduces overhead. The calculation of line crossings (and crossings angles) is the most computationally expensive, with a complexity of $O(h^4)$ or $O(n^2)$ (where h is the size of the display, see Section 3.2). All others are $O(h^2)$ or $O(h)$, with $h << n$ (i.e., $O(n)$ would in many cases be more expensive in real-world terms).

The optimization requires the calculation of the relevant metrics (which the user picks) for all axis pairs. They are used to build a cost matrix, which the branch-and-bound optimizer needs for building candidate solutions. Once the matrix is built, the optimization is very efficient. For a dataset with 12 dimensions, it typically queues several ten thousand partial solutions, but only evaluates a few (less than ten) complete ones. That means that our culling criterion is very good, and that the best-first search has a high chance of immediately finding the best overall solution.

Multiple metrics are handled in the matrix building stage, and have no direct bearing on the performance of the optimization itself. Having to evaluate several metrics of course slows down the matrix construction. Axis inversions similarly are handled at the first stage and do not increase the complexity of the search space. Since our implementation efficiently computes the inverted value together with the non-inverted value for most metrics, the added cost for considering inversions is very low.

6 Discussion

While Scagnostics are based on a projected data space, the source data for Pargnostics is a sampled, binned space that corresponds directly to the pixels on the screen. This makes Pargnostics inherently scale-dependent, and thus mirror what the user actually sees when working with an interactive visualization. Like Scagnostics metrics, our proposed metrics are not orthogonal to each other, but have dependencies between them. In parallel coordinates, axis inversions have a direct impact on the structures one sees on the screen. Although this adds to the complexity of the search process for the end user, he has more ways to search the visual space for useful structures. Some of the structures which may not be obvious in non-inverted axes may become obvious due to inversion of the axes.

The structures we quantify can be described in terms of more than one metric taken together. The quality of the rendered image can be described both in terms of over-plotting and pixel-based entropy. Clutter can be quantified in terms of a high number of crossings and low crossing angles; high parallelism and low crossing angles tend to produce clusters; low crossings usually lead to high parallelism, but may also be manifested in converging/diverging structures with lines clustered close to each other. Correlation between dimensions can be quantified in terms of parallelism and mutual information. Since combining metrics can reveal interesting structures, we allow the user to combine multiple metrics in the optimization.

One limitation of our approach is that it we have cannot be sure that our set of metrics is exhaustive, and can only test their effectiveness in case and user studies. The metrics also require knowledge of the user's screen and display size, and need to be recalculated when the display is resized.

7 Case Study

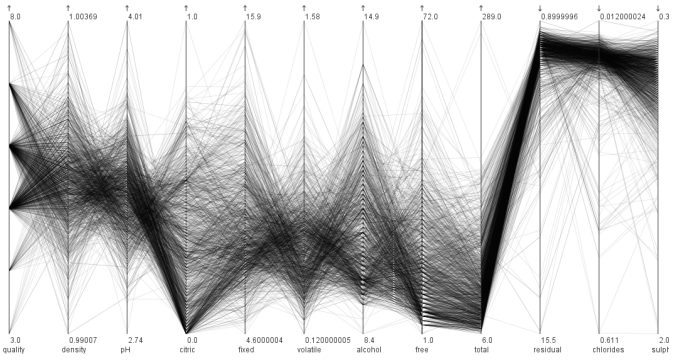
We demonstrate the use of our metrics with two example datasets, one describing car models and another one about wines.

7.1 Cars Dataset

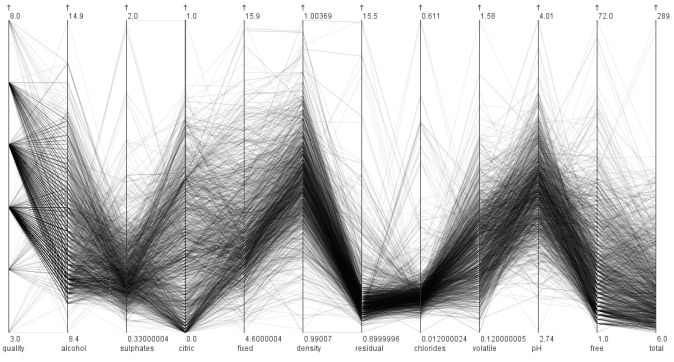
The cars dataset [26] consists of 392 values and six attributes: *MPG*, *horsepower*, *cylinders*, *weight*, *acceleration*, and *year*. Pargnostics metrics can be used for both user-centered and automated optimization. Based on the common scatterplot matrix, we have developed a parallel coordinates matrix view that shows all combinations of axes in the lower left half of the matrix with both axes pointing in the same direction, and with inverted axes in the upper right (Figure 8). A similar matrix view shows the distance and angle histograms for each axis pair (Figure 9).

The small parallel coordinate plots show the overall structure of the axes pairs, while the histograms give some insight into some of the metrics. In our prototype implementation, the user can construct the parallel coordinates display by picking axis pairs from any of the matrices. There is also a ranked view of parallel coordinate axis pairs (Figure 10), which allows the user to directly find parts of the display that exhibit certain structures.

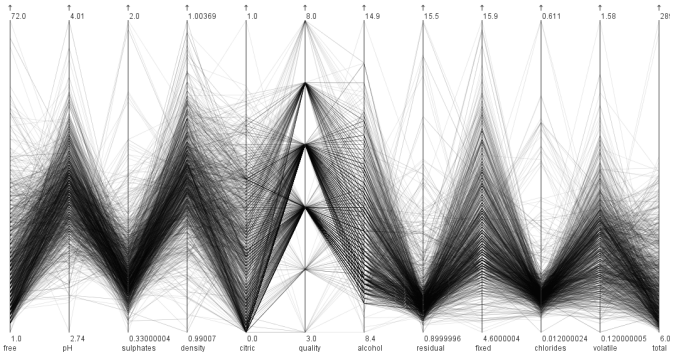
Selecting from any of those views is based on visual structure rather than particular data dimensions, and thus frees the user from preconceived ideas about the data. Adding dimensions is typically done by selecting them by name; in our model, the display can be built directly from interesting visual structures. The resulting display makes maximum use of the power of the visualization.



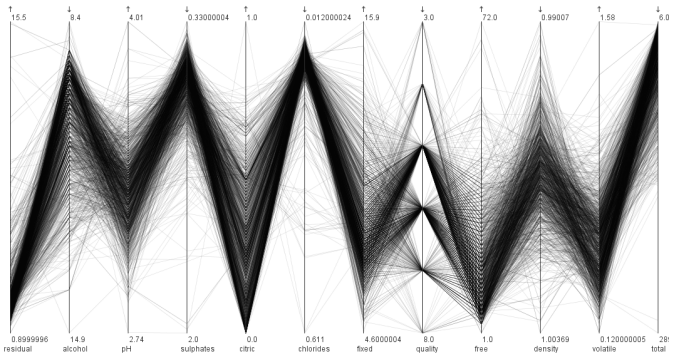
(a) View with maximized angles of crossing, including axis inversions (last three axes).



(b) Minimizing the number of crossings.



(c) Minimized angles of crossing and maximum parallelism.



(d) Maximized number of crossings and minimized angles of crossing, and including inversions.

Fig. 11. Optimization for different sets of criteria using the wine dataset.

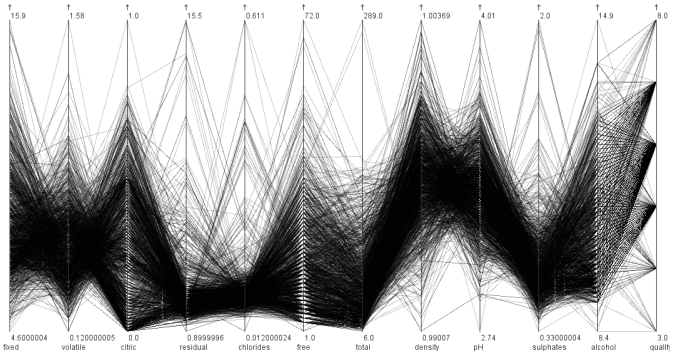


Fig. 12. Initial layout of the wine dataset.

7.2 Wine Dataset

The wine quality dataset [26] consists of 4898 rows and 12 dimensions: *fixed acidity*, *volatile acidity*, *citric acid*, *residual sugar*, *chlorides*, *free sulfur dioxide*, *total sulfur dioxide*, *density pH*, *sulphates*, *alcohol*, and *quality*. Figure 12 shows the initial view of the data set, with the dimensions in the order in which they appear in the data file.

7.2.1 Single-Metric Optimization

In Figure 11(a), parallel coordinates is optimized by high crossing angles, taking inversions into account. As we can see, most of the lines between the axes tend to cross at close to 90 degrees and this helps to reduce clutter. In Figure 11(b), parallel coordinates is optimized by minimum number of crossings. This tends to produce a less cluttered display, but at the same time produces high parallelism as observed on *density* and all dimensions to the right of it.

7.2.2 Multi-Metric Optimization

In Figure 11(c), parallel coordinates is optimized by low crossing angles and high parallelism. This configuration produces clusters on either side of the categorical quality variable. Thus this configuration is not only useful to see the correlations and clusters, but also enables one to see where the intense concentration of records, or the mode [29], occurs. In Figure 11(d), parallel coordinates is optimized for high number of crossings and low crossing angles, taking axis inversions into account. There are many inverse correlations observed between most of the axes. A large number of crossings generally tends to produce inverse correlated structures and with low angles, the clusters can be seen clearly.

8 Conclusions and Future Work

Pargnostics metrics provide a new way of looking at parallel coordinates. We argue that defining parallel coordinates in terms of small multiples, i.e., axis pairs, enables us to optimize the display in an efficient manner. Screen-space metrics not only describe the rendered image on screen, but also quantify the different visual structures one can see. In this sense, Pargnostics fills a gap in the existing literature on parallel coordinates. Being able to analyze what ends up on the screen makes it possible to provide better visualization setups that take the specific properties of the visualization technique into account.

We are expanding this work to cover more visualization techniques, and adding additional metrics as needed. Some of the current metrics also apply to other techniques, but additional ones are certainly necessary. We are also working on devising user studies to test the effectiveness of our metrics, and to see how well users will be able to handle them directly.

In addition to the optimization work shown in this paper, we want to use our metrics to better understand the loss of information when visualizing it, which is a useful property of visualizations when the data cannot be shown at full resolution for privacy reasons.

REFERENCES

[1] J. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26:832–843, 1983.

[2] R. Amar, J. Eagan, and J. Stasko. Low-level components of analytic activity in information visualization. *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, pages 111–117, 2004.

[3] G. Andrienko and N. Andrienko. Constructing parallel coordinates plot for problem solving. *1st International Symposium on*, pp:9–14, 2001.

[4] M. Ankerst, S. Berchtold, and D. Keim. Similarity clustering of dimensions for an enhanced visualization of multidimensional data. In *Proceedings Information Visualization*, page 52. IEEE CS Press, 1998.

[5] E. Bertini and G. Santucci. By chance is not enough: Preserving relative density through non uniform sampling. *Information Visualisation, International Conference on*, 2004.

[6] E. Bertini and G. Santucci. Visual quality metrics. In *BELIV '06: Proceedings of the 2006 AVI workshop on BEyond time and errors*, pages 1–5, New York, NY, USA, 2006. ACM.

[7] M. Chen, D. Ebert, H. Hagen, R. Laramee, R. Van Liere, K. Ma, W. Ribarsky, G. Scheuermann, and D. Silver. Data, Information, and Knowledge in Visualization. *IEEE Computer Graphics and Applications*, 29(1):12–19, 2009.

[8] T. Cover and J. Thomas. Elements of information theory. *Wiley*, 2006.

[9] G. Ellis and A. Dix. Enabling automatic clutter reduction in parallel coordinate plots. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):717–724, 2006.

[10] C. Forsell and J. Johansson. Task-based evaluation of multirelational 3D and standard 2D parallel coordinates. *Proceedings of SPIE*, pages 64950C–64950C–12, 2007.

[11] W. Huang, S.-H. Hong, and P. Eades. Effects of Crossing Angles. In *Proceedings Pacific Visualization Symposium*, pages 41–46. IEEE CS Press, 2008.

[12] A. Inselberg. Multidimensional detective. In *Proceedings Visualization*, pages 100–107. IEEE CS Press, 1997.

[13] A. Inselberg. *Parallel Coordinates: Visual Multidimensional Geometry and Its Applications*. Springer, 2009.

[14] A. Inselberg and B. Dimsdale. Parallel coordinates: A tool for visualizing multi-dimensional geometry. In *IEEE Visualization*, pages 361–378. IEEE CS Press, 1990.

[15] J. Johansson and M. Cooper. A screen space quality method for data abstraction. *Comput. Graph. Forum*, 27(3):1039–1046, 2008.

[16] D. Keim. Designing pixel-oriented visualization techniques: theory and applications. *IEEE Transactions on Visualization and Computer Graphics*, 6:59–78, 2000.

[17] J. Li, J.-B. Martens, and J. J. van Wijk. Judging correlation from scatterplots and parallel coordinate plots. *Information Visualization*, 9(1):13–30, 2010.

[18] M. Lind, J. Johansson, and M. Cooper. Many-to-Many Relational Parallel Coordinates Displays. *2009 13th International Conference Information Visualisation*, pages 25–31, July 2009.

[19] N. Miller, B. Hetzler, G. Nakamura, and P. Whitney. The need for metrics in visual information analysis. In *NPIV '97: Proceedings of the 1997 workshop on New paradigms in information visualization and manipulation*, pages 24–28, New York, NY, USA, 1997. ACM.

[20] J.-F. Rit. Propagating temporal constraints for scheduling. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 383–388, 1986.

[21] J. Schneidewind, M. Sips, and D. A. Keim. Pixnostics: Towards measuring the value of visualization. In *Proceedings Visual Analytics Science and Technology*, pages 199–206. IEEE CS Press, 2006.

[22] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings Visual Languages*, pages 336–343. IEEE CS Press, 1996.

[23] A. Tatu, G. Albuquerque, M. Eisemann, H. Theisel, M. Magnor, and D. Keim. Combining automated analysis and visualization techniques for effective exploration of high-dimensional data. *IEEE Symposium on Visual Analytics Science and Technology*, pages 59–66, 2009.

[24] E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 2nd edition, 2001.

[25] J. Tukey and P. Tukey. Computing graphics and exploratory data analysis: An introduction. In Proceedings of the Sixth Annual Conference and Exposition: Computer Graphics 85. In *Proceedings of the Sixth Annual Conference and Exposition: Computer Graphics*, pages 773–785, 1985.

[26] UC Irvine Machine Learning Repository. http://archive.ics.uci.edu/ml/.

[27] P. Vázquez, M. Feixas, M. Sbert, and W. Heidrich. Viewpoint selection using viewpoint entropy. In *Proceedings Vision, Modeling, and Visualization*, pages 273–280, 2001.

[28] C. Ware, H. Purchase, L. Colpoys, and M. McGill. Cognitive measurements of graph aesthetics. *Information Visualization*, 1(2):103–110, 2002.

[29] E. Wegman. Hyperdimensional data analysis using parallel coordinates. *Journal of the American Statistical Association*, 85, 1990.

[30] L. Wilkinson, A. Anand, and R. Grossman. Graph-theoretic scagnostics. In *Proceedings Information Visualization*, pages 157–164. IEEE CS Press, 2005.

[31] J. Yang, M. Ward, E. Rundensteiner, and S. Huang. Visual hierarchical dimension reduction for exploration of high dimensional datasets. In *Proceedings Data Visualization*, pages 19–28. Eurographics Press, 2003.

[32] H. Zhou, X. Yuan, H. Qu, W. Cui, and B. Chen. Visual clustering in parallel coordinates. *Computer Graphics Forum*, 27(3):1047–1054, 2008.