Contact author:

Greg Ross
Department of Computing Science
University of Glasgow
Lilybank Gardens
Glasgow
G12 8QQ
UK

Tel: +44 141 3303339
Fax: +44 141 3304913

gr@dcs.gla.ac.uk

Coordinating Components for Visualisation and Algorithmic Profiling

Greg Ross, Alistair Morrison, Matthew Chalmers
University of Glasgow
{gr, morrisaj, matthew}@dcs.gla.ac.uk

Abstract

A number of researchers have put forward approaches to the development and use of visualisation systems consisting of a number of components, through which data and interaction commands flow. Systems based on hybrid and multistage algorithms can be used to reduce algorithmic complexity, and to open up intermediate stages of the algorithm for inspection and steering. In this paper we present work on aiding the developer and the user of such algorithms, applying interactive visualisation techniques to the process of designing, evaluating and using visualisation systems. We present a set of tools designed to show and control the performance of other visualisation components, and we offer case studies of their application to a number of data sets. Through this work we are exploring ways in which techniques traditionally used to prepare for visualisation runs, and to retrospectively analyse them, can find new uses within the context of a multi–component visualisation system. We aim to demonstrate that when such systems use flexible structures for data flow and cross–component interaction, developers and users can gain valuable understanding and control of the processes and parameters of visualisation, and hence insight into the information being visualised.

Keywords--- **Visual programming, data-flow model, multiple views, coordinated views, hybrid algorithms, dimension reduction, algorithmic profiling.**

1. Introduction

The challenge of gaining insight into the information contained within a set of multidimensional data entails finding a representation of the data that conveys inherent patterns and latent structure. A dimension reduction approach to this problem seeks to represent multidimensional data in a low dimensional space, such that inter-object distances are preserved as well as possible. There have been many layout algorithms proposed for effectively reducing data dimensionality [1, 2, 3, 4], each with different benefits and drawbacks. For example, some techniques may be too time-consuming to be able to process any sizable data set, whereas others may not be flexible enough to reveal complex and diverse relationships within the data. A combinatorial or 'hybrid' approach to building dimension reduction solutions has been shown to provide good results in terms of efficiency and the uncovering of patterns contained within complex data [5].

A software system called HIVE [6] has been developed that provides a framework to encapsulate this approach. The benefits of hybrid layout algorithms are twofold; as well as providing fast, effective solutions, the approach also naturally provides multiple views of a data set as it is transformed in the various algorithmic stages.

HIVE provides an extensible palette of algorithmic components that can either be used individually or integrated into hybrid algorithmic models. The system is designed as an environment in which to design such hybrid models. As such, this paper introduces a new suite of tools that permit the evaluation of these models within the HIVE system, as well as offering further functionality for the exploration of high dimensional data sets

Due to the flexibility of the visual programming approach, these tools can be connected to various stages of a hybrid algorithm whilst it is running, using visual metaphors consistent with the rest of the HIVE framework, and as such they themselves become visualisations. By coordinating the views provided by these tools with the views of the data, we can not only interactively explore our data but also the relationship between the layout and the performance of the transformation process as it runs.

In the following section, more background information is provided on the HIVE framework and hybrid algorithms in general. Following that, the new profiling modules are described and then three case studies are presented that illustrate their benefit. Finally, a section of future work precedes the presentation of our conclusions.

2. Background and previous work

This section provides a more detailed overview of the HIVE system, after a general introduction to hybrid layout algorithms and the multiple views that such an approach provides.

2.1 Hybrid algorithms and coordinated views

Several approaches exist for tackling dimensional reduction, each with different benefits and drawbacks. While linear projection approaches such as PCA can be fast to compute, they can only reflect patterns in the data that can be explained by a linear function of the dimensions, and thus do not reveal non-linear relationships. Conversely, methods such as force-directed placement (FDP) [3, 7] create layouts through the modelling of physical forces based upon inter-object relationships. Such algorithms can be used to reveal such non-linear relationships, but until recently exhibited high time complexity.

It has been shown that the adoption of a hybrid (or multi-stage) approach to the creation of layout algorithms can help to reduce time complexity and aid data exploration [5, 8, 9, 10]. Based upon trade-offs between individual algorithms such as those described above, individual models may be combined or tailored in such a way as to maximise the benefits of each, whilst limiting any shortcomings. For example, the hybrid algorithm presented in [10] applies force-directed placement to a random sample of the data, and then uses the resultant layout as a base on which to interpolate the rest of the set. In doing so, the computationally intensive process is applied to only a small subset, but we maximise the benefits gained from the accurate positioning of this subset in positioning the remaining objects.

Another benefit of this hybrid approach is the provision of intermediate views of the data. Hybrid algorithms may be implemented as a flow of data through a series of computational components. As such, the output of each intermediate stage may be visualised to provide further insight into the data under investigation. In [9] a two-phase algorithm is proposed where a self organising map (SOM) [4] – an artificial neural network, usually visualised as a discrete grid of glyphs that visually encode the neural weights – is used to create a topological layout of data, and also used as the input for force-directed placement. In this case, the SOM can provide a discretised overview of the data while the FDP output provides a view where local detail is better defined.

The hybrid algorithmic approach can potentially result in multiple views of the data as it is transformed, and therefore would benefit from some form of view coordination. In North and Shneiderman's snap-together system [11], it is shown that users can direct coordinations between different views of data, thereby gaining more information. In [12] a model for view coordination is described, in which subsets of interactive components can be connected to support brushing and linking, and data flow specifications can be used to control the running of components, e.g. an FDP component triggering 'downstream' components only when it has stabilised. If users and designers of hybrid layout algorithms can supplement their views with such coordination functionality, the effects of individual algorithmic stages can be gauged and the data explored within different transformation contexts. An example of this is given in [6] where significant local detail within a data set was visible only in a view created by an intermediate step of the layout process. In this case, the intermediate view was also coordinated with a histogram and fisheye table view to gain more insight.

2.2 The HIVE framework

Visual programming can be described as the adoption of a 2D graphical notation for the creation of programs, as opposed to the conventional 1D textual approach [14]. The implementation of visual programming systems is often based on the data-flow model where functional components are directly manipulated by the user to create a block diagram representing the processes through which data flow. This provides an intuitive and flexible basis for the creation of customised computing applications without the need for any 'real' programming expertise.

HIVE (Hybrid Information Visualisation Environment) [6] is a data-flow visual programming system developed to examine and create clustering and layout algorithms, as well as to explore the data that they transform. HIVE was inspired by systems such as Upson et al's AVS [13], which utilises the data-flow model and visual programming to interactively create scientific visualisation applications, and North and Shneiderman's snap-together software [11], where the user can define view coordination for interaction flow

The HIVE system uses direct manipulation to allow users to interactively create and explore hybrid layout algorithms. Visual programming and a novel algorithmic architecture allow users to semi–automatically coordinate multiple views and interactively steer data flows. The architecture has been designed so that users can compose exploratory data analysis tools using modular components for importing data, algorithmic processing and graphical rendering.

Figure 1 shows an example of HIVE's workspace where a simple data-flow network was built manually by dragging the appropriate component modules from a GUI tool bar[1]. Each module has several ports, and by dragging connections between these, the components may be 'wired' together. Labelled ports are made visible in HIVE's 'link' mode.

The data set used in the figure is in the form of a file containing 300 two-dimensional coordinates representing a 'C' shape (a square missing one side). The data module is connected to an FDP module and a fisheye table, which uses distortion to provide focus and context information in a similar way to Table Lens [15]. Note that the data is fed into the FDP and table modules simultaneously. In the figure, the FDP process has completed, although it may be noted that during the

[1] Throughout the paper, certain details have been omitted from screenshots to aid clarity.

iterative process, the scatterplot of the 2D layout coordinates was updated every ten iterations. The scatterplot therefore displays an animation of the layout process, allowing the user to watch the layout form. The link that is highlighted in red (link between table and scatter plot) is a coordination link. This means that by selecting rows on the table or points in the scatter plot, the corresponding items are highlighted in the other view. This location probing is an example of the flow of interaction possible within the system.

HIVE has a novel hybrid algorithmic framework, offering a general approach to the composition of efficient and flexible hybrid algorithms. The choice of each algorithmic component is influenced by many characteristics including the cardinality, dimensionality and distribution of the data, computational cost and the interaction components that might be used within a larger workspace. These choices can be made incrementally, so that users may employ intermediate representations as they work with and explore the data.

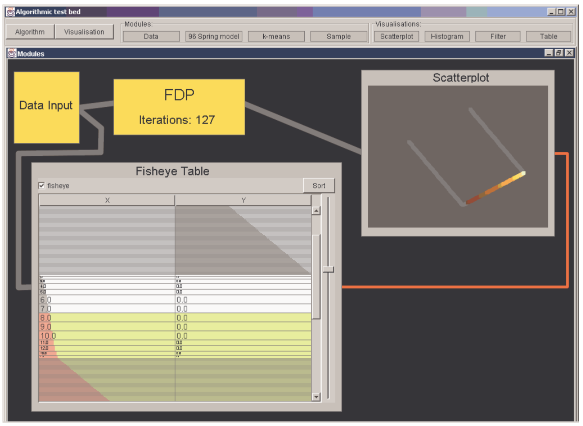


Figure 1. A simple example of a data flow network in HIVE. Synthetic input data is processed by a force-directed placement (FDP) routine while being viewed in a fisheye table. The output of the FDP is fed into a 2D scatterplot. The two visualisation components are coordinated, so that selecting objects in one view will highlight the corresponding objects in the other.

HIVE can also assist the user by using a pre-authored classification of data—based on cardinality and dimensionality of data sets—and a corresponding classification of available algorithmic components based on the classes of input data set for which each component is suited, and the class of data set it outputs. This means that if a data file is loaded, and a visualisation component selected, the system will use its 'cookbook' of algorithms to set up a number of components to take the data file's contents and create data suitable for visualisation, e.g. reading in a large high–dimensional data set and choosing an appropriate dimensional reduction algorithm to create 2D data suitable for a scatterplot.

While the system provides flexibility in the creation of layout algorithms and data exploration, there is also a requirement to be able to assess the quality of the layouts produced. In some instances, a new layout algorithm might produce a view of a data set in which there is no discernable structure. However, this may or may not be representative of the data and therefore we require some means of determining how well inter-object distances in the layout are preserved. In Buja et al's XGvis software [16], the user is able to experiment with different hard-coded MDS algorithms. The software allows the user to change the algorithmic parameters such as the dimensionality of the output space, and to quantitatively measure the effects of such changes on the quality of the representation. Similar techniques for algorithmic steering and profiling have been incorporated and given a more interactive role within HIVE, as discussed in the following sections.

3. Profiling modules

HIVE has been demonstrated to be an effective environment within which to explore high dimensional data sets. A palette of algorithmic components and visualisation tools provides the user with several disparate views of a data set and allows a number of different aspects to be explored. Further insight is supported via brushing between these views. The framework is also suitable as an environment for the design of novel hybrid algorithms. Novel combinations of implemented modules may be experimented with, and the extensible nature of the algorithmic palette permits the simple addition of new components.

In addition to this, we propose that HIVE is a useful tool for the profiling and evaluation of hybrid algorithms. A number of HIVE modules have been implemented to measure and display performance characteristics of other HIVE modules. The inclusion of such profiling modules permits algorithm evaluation to be tightly and interactively coupled with the algorithms being run. The same visual metaphors may be used in linking together profiling components, and the decision as to which properties to measure can be made and altered at run-time. Profiling tools may also be linked to existing visual modules, with their coordinated use providing insight into data sets that would go unnoticed in a sole visualisation. Examples of such coordination are provided in section 3.

In this section, we introduce the profiling modules implemented in HIVE.

3.1. Multiple runs module

In evaluating hybrid algorithms, 'batch-runs' of algorithmic executions need to be performed. Fundamental to such a system of algorithmic profiling is a controller to coordinate the execution of each test run.

The Multiple Runs (MR) module fulfils this role. As all modules in HIVE, the MR module interacts with other components via a series of signals sent through connected ports. MR has three output ports: a 'data out' port through which to pass the input data to algorithmic components, a start trigger to activate the first module, and an optional parameter port to provide a mechanism for customising the conditions under which each test should be run. For example, it is possible to customise the use of an FDP component on each run by supplying values for the number of iterations to perform, the level of damping to apply to the model, and the values for other constants used within the algorithm. It is necessary to specify algorithmic conditions in such a manner, rather than via interactive controls on each component, so that a number of different conditions can be experimented with during a batch job.

Such parameters are input, along with information on number of runs and which datasets to use, in a text entry field within the module. Series of commands are entered in the form:

(DataFile,[NumRuns,<ModuleID,(parameters)>,<>…])

This input is parsed by the MR module, stripping out parameter information and passing it to the modules in the form of a series of (moduleID,(parameters)) tuples. Each module receives the entire parameter list, and must search it for the appropriate entry. On the termination of an algorithm, the final module sends a signal to the MR's input port, and the MR begins the next run.

3.2. Clock

Run-time is an important criterion in the evaluation of a layout algorithm. Regardless of the effectiveness of a particular technique, excessively high run times may render its use infeasible, or place a limit on the size of data set which can be laid out within a time suitable for interactive use. It is obviously also useful to be able to compare algorithms' run times against those of competitor techniques.

To this end, a clock module has been defined with which a developer may easily measure the run times of one or more algorithmic components. Figure 2 illustrates how the clock may be used to measure the time taken for individual stages of a hybrid model. One clock module is connected to the first stage, and another to the second. In the HIVE environment, it is simple to use the clock to time a complete algorithm, by connecting the first algorithmic component to the *start* trigger and the final module to the *stop* trigger.

On completion, the run time is displayed on the clock module and also written to a file, the name of which may be specified within a text entry box. Multiple algorithmic runs may therefore be executed, with each clock appending to a separate results file each time.

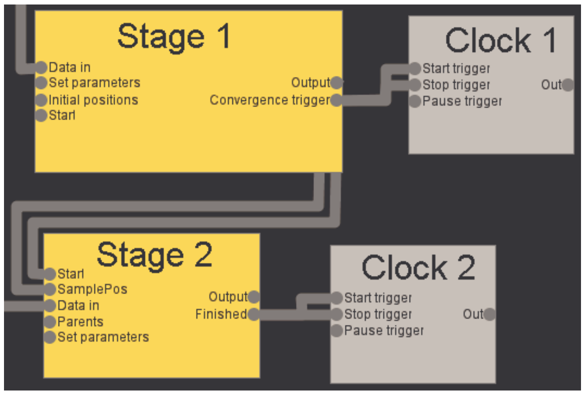


Figure 2. Clock modules connected to each stage of a hybrid algorithm. HIVE's extensibility allows multiple instances of profiling modules to be connected at run-time. In this screenshot, HIVE is in 'link' mode, meaning that labelled ports are displayed while other controls are temporarily hidden.

3.3. Stress

In evaluating a layout algorithm, as well as examining run times, it is of obvious importance to consider the quality of layouts produced. Several metrics exist for the assessment of layout quality [17, 18, 19], often based on the layout's *stress*: the discrepancy between layout distances and high dimensional relationships. The stress-1 metric [17], for example, is defined as in the equation below, where *h* represents high-dimensional distance and *l* low-dimensional (layout) distance:

$$Stress = \frac{\sum_{i<j}(h_{ij} - l_{ij})^2}{\sum_{i<j} l_{ij}^{2}} \tag{1}$$

A stress module has been defined to perform such evaluations. Taking from input ports a high dimensional data set and a set of low-dimensional positions, the above stress calculation may be performed. We choose to receive data in this form, rather than sets of inter-object distances for issues of space efficiency. The high and low-dimensional distances are calculated on the fly within the stress module. In common with the clock module, a trigger port can be used to commence stress calculation. A button is also supplied to give the instruction to calculate the stress immediately. Although redundant in batch-job execution, such functionality is important to interactive exploratory analysis of data.

Calculation of stress is a computationally intensive undertaking, often a more time-consuming process than the algorithm itself. Although useful in assessing layout quality, it may not be integral to the algorithmic process. The time spent on stress calculations should not, therefore, contribute to run time measurements. As such, a 'pause' output port is supplied on the stress module.

Any clock to which this port is connected will be instructed to stop timing for the duration of stress calculations, and the time spent on them will not contribute to the measured run times.

Like the clock module, the stress component has a text field to allow users to specify an output file to which results should be written. Controls are also provided to allow selection between several implemented stress algorithms.

3.4. Shepard diagram

The Shepard diagram [2] is another tool that may be used to determine the quality of an MDS solution. At its most basic, it is a plot of the low dimensional (or layout) distance between each pair of objects in the data set against the corresponding high dimensional (or input) distance for that pair. Should an MDS solution provide a good representation of relationships within the high dimensional space, the points in the Shepard diagram will have a strong positive correlation with low deviation from the 45 degree diagonal. Figure 3 illustrates a Shepard diagram, where high dimensional distances are plotted on the y-axis and layout distances along the x-axis. Points that lie above the diagonal represent data elements that are too close together in the generated layout, while points that lie below the diagonal represent those that have been positioned too far apart. It is clear that points distant from the diagonal in such a plot would represent pairs of objects that make a large contribution to layout stress.

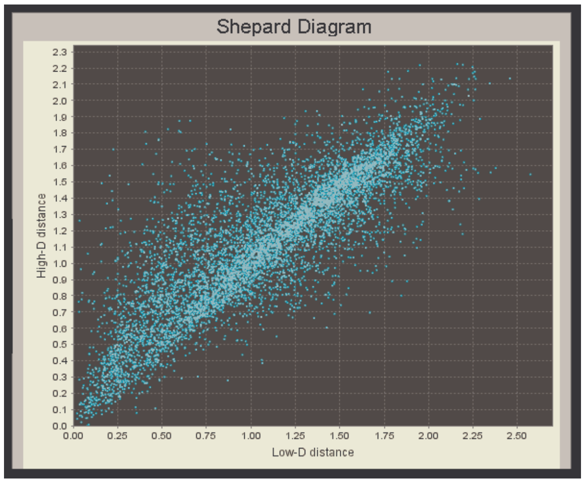


Figure 3. The Shepard diagram module. Each point represents distances between a pair of objects: high dimensional distances (y-axis) are plotted against low dimensional layout distances.

As with obtaining the stress of a layout, acquiring a full Shepard diagram necessitates the calculation of the distances between all pairs of elements of the data set, and this can be very time consuming. To overcome this, like the stress module, the Shepard diagram module allows the user to specify the size of a random sample of objects with which to produce the plot, of default sample size $\sqrt{N}$. Brush and link functionality is also provided so that location probing in the Shepard diagram can highlight the corresponding data items represented in other views.

The Shepard diagram has a long tradition of use in MDS as a tool for the appraisal of layout quality. Historically, MDS was performed by psychologists and psychophysicists [19], taking as input experimental subjects' judgements of similarity between a set of stimuli. As such, data sets tended to be small: at most a few hundred objects. The Shepard diagram could therefore comfortably cope with the $\frac{1}{2}N(N-1)$ distances between N objects. In more recent times, such techniques have been used for exploratory data analysis of much larger data sets. It is not uncommon for a user to explore a set of 100,000 objects or more. As such, the Shepard diagram became a less useful tool, due to the density of points required for such sets.

In an environment such as HIVE, however, the provision of interactive tools allows us to find new uses for such traditional tools. Zoom and pan controls, in conjunction with sampling, can help alleviate the usability problems encountered by overly dense plots. Additionally, the interactive coordination of several plots or views of data can promote understanding into data that would not be possible with the solo techniques.

3.5. Chart

A general line chart tool has also been added to the HIVE environment, and is illustrated in Figure 4. Dependent and independent variables may be specified via separate ports, with a line chart drawn dynamically as values are received. Multiple data series may be accommodated, with each measurement being assigned to a series via parameters or interactively.

Although simple, such a tool has many practical applications. For example, algorithmic characteristics such as run time or stress can be graphed, allowing comparisons to be made between various models or algorithm stages. The chart can also be used dynamically during algorithm execution, for example to view the stress of a layout over time. Such usage has been illustrated in Figure 4. The chart's y-axis represents layout stress, while the x-axis counts the number of iterations performed. Such a graph could be used in detecting situations where an iterative model has become stuck in a local minimum. Functionality is also provided to support the simple export of the chart as an image file.

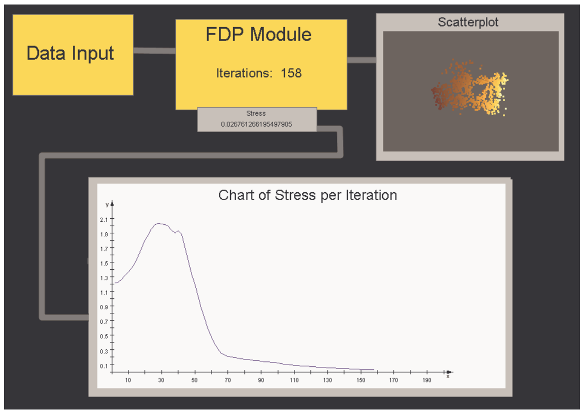


Figure 4. The Chart component being used in conjunction with an FDP routine to display stress per iteration. Note that stress initially grows as energy is put into the system, before dropping as the layout progressively improves. The scatterplot visualisation shows the layout after 158 iterations. Using the two views in parallel, a user can determine whether the layout is complete, or further processing is required.

3.6. Coordination of profiling modules in HIVE

The previous sections have described isolated instances of HIVE's new profiling modules. The real power of such techniques, however, comes in their combination and interaction with existing components within the HIVE environment. Histograms, fisheye tables and scatterplots all have interactive functionality, allowing coordination with the profiling components.

For example, the Shepard diagram is traditionally a static presentation technique for qualitatively evaluating the quality of a low-dimensional representation of high-dimensional data. By incorporating it into the HIVE framework, the plot can have as many instantiations as necessary, with each instance connected to a different part of the visualisation's data flow. By connecting a Shepard diagram to a scatterplot, we create an interactive link between the two plots. Making a selection in the Shepard diagram will therefore highlight those objects in the scatterplot layout whose pairwise distances correspond to the selected points. For example, in Figure 5, we use Shepard diagrams to compare layouts obtained from PCA and FDP. It is apparent that the Shepard diagram of the PCA layout has a distinct diagonal edge, below which no points are plotted. This may be explained by the fact that PCA functions via a projection of the high dimensional space onto a 2D plane. In contrast, FDP has no linear constraints and attempts to position objects to best preserve high-dimensional relationships. This results in a Shepard diagram where there are points both above and below the diagonal, as well as the points that represent items ideally placed in the plane.

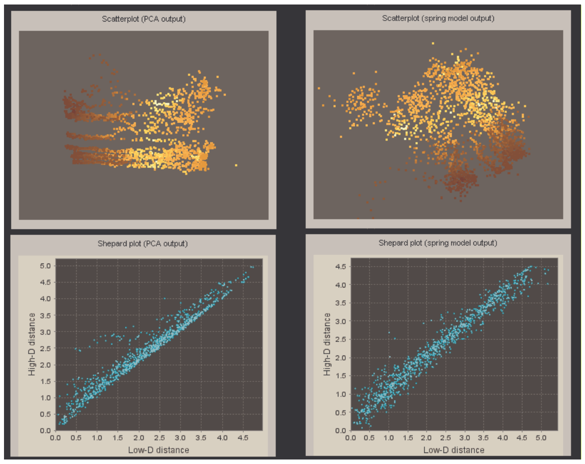


Figure 5. The top two images show 2D layouts of the Antarctica data (described in section 4.2.2) using linear PCA (left) and nonlinear FDP (right). Below each layout is the associated Shepard diagram. The PCA layout has no pair of objects at a greater distance from each other than in high-dimensional space. This is confirmed by the fact that no objects appear below the diagonal in the PCA's Shepard diagram.

Another implemented component that may be used in interactive combination with the profiling modules is a module for Voronoi tessellation. This module may be used to partition a completed layout [20]. Each point is contained by a convex polygon so that the portion of space contained within the polygon is closer to that point than any other. Clustering may then be performed by finding contiguous groups of polygons where the density of points is similar. The module may be used in combination with profiling modules to detect clusters that may benefit from closer examination.

In the following section, we provide more concrete examples of the coordination of components with a series of case studies.

4. Case studies

This section details several case studies, illustrating the coordinated use of existing components with the novel profiling modules within the HIVE environment.

4.1. Batch job of executions for algorithm evaluation

The first case we will examine is the thorough evaluation of a novel algorithm. Through the use of the profiling modules described in the previous section, such evaluation may be performed simply and in an intuitive manner. The following describes the evaluation process undertaken in the writing of [10]: a presentation of a novel hybrid layout algorithm.

In performing such algorithmic profiling, a large number of executions are necessary. Several models may be evaluated on several different data sets of several different sizes. In addition, results should be averaged over numerous runs: an especially important consideration in the case of iterative models such as that proposed in [10], which can occasionally become stuck in local minima. With several hundred executions necessary for a thorough evaluation, then, it is clear that an automated profiling process is a useful aid to the designer. Figure 6 illustrates the configuration of components required for such an evaluation. To avoid describing the specific model in depth, details such as the names of individual components have been omitted from the figure.

Having built a hybrid algorithm (composed of the modules shaded in yellow in Figure 6), we wish to examine its performance in comparison with an alternative technique. Profiling modules (grey) may be added to the module configuration at the user's discretion. Here, we have elected to measure the run time of two stages. We also measure stress at the conclusion of stage 4. By specifying file names on each of these components, separate output files are generated by each, allowing the detail of performance characteristics to be explored further in other applications.

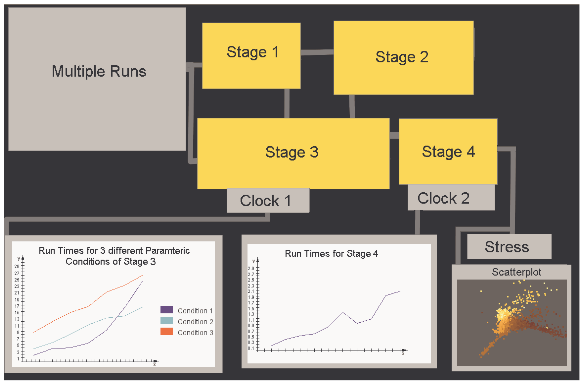


Figure 6. The yellow modules represent different stages of a hybrid algorithm. The Multiple Runs module coordinates a sequence of executions, loading data and parameters into each component. Charts graph run time over data set size at various stages of the algorithm. The bottom left chart shows run times under 3 separate sets of parameters for stage 3. Having connected the various components and provided instructions to the MR module, the algorithm executions and chart plotting may proceed unsupervised.

The MR module (top left) is provided with a list of data files and parameter commands. It systematically loads in each file, and passes the instructions for the current execution to each algorithm component. The MR module passes a start trigger into stage 1, and receives another from the final stage to indicate that the algorithm has terminated: the cue to reset all modules and begin another run.

Finally, charts at the bottom of the figure display experimental results. A test data collection has been sampled to create data sets of varying size. The charts graph run times against data set sizes under different algorithmic conditions, with each line on the chart representing a different condition. Each chart is connected to a different clock module, and therefore displays times taken by different algorithmic components.

For example, the chart at the left hand side displays the run times required by the third stage of the model. Three separate approaches were experimented with for this stage (as specified by the experimenter in the MR module and passed to stage 3 via parameters), as indicated by the 3 lines on the chart. We can deduce that on small data sets, condition 1 executed the fastest, whereas condition 2 becomes optimal as data size increases.

Charts of this type formed the basis of the results section of [10]. What could have been a laborious evaluation procedure was undertaken via a simple, unsupervised process. The algorithms could be left to run overnight, with the generated charts being exported in graphics format as they were created.

4.2. Exploratory analysis

This section demonstrates via two examples how the profiling modules may be used interactively to further understanding of a data set.

4.2.1. Synthetic data. In this example, we illustrate the interactive combination of the Shepard diagram with other HIVE visualisations. As previously mentioned, brush-and-link coordination has been incorporated in the Shepard diagram to allow interaction with other components. For example, linking the Shepard diagram and scatterplot views allows insights into the relationships between quality of positioning and objects' placement within the layout.

To illustrate the usefulness of this interactive capability, we provide an example using a synthetic data set representing a 3D cube. Such a data set is a useful test case, as it is impossible to represent perfectly in a 2D space, and no 2D projection of the data is much better or worse than any other. To begin, we use PCA to obtain a scatterplot layout (Figure 7a). Linear projection-based layout techniques such as PCA, although fast, provide a layout based upon global data properties. It is therefore the case that certain local area may be especially poorly represented. This example illustrates how interactive use of the Shepard diagram component can help a user to resolve inaccuracies in these areas, and thereby enhance understanding of the structure of the data.

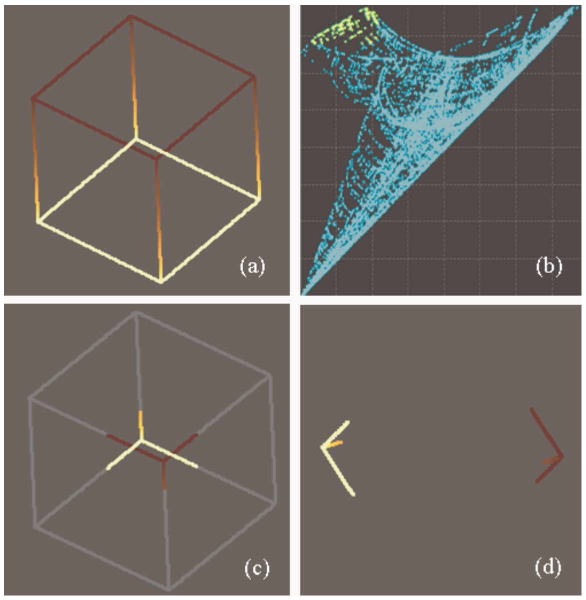


Figure 7. PCA layout and Shepard diagram working together interactively to help build user understanding of a data set. (a) The PCA layout. (b) A Shepard diagram of the layout. A selection is made of points corresponding to distances in the layout that may benefit from further analysis (highlighted region). (c) The selection in the Shepard diagram is also highlighted in the PCA layout. (d) A re-projection of the selected points and their immediate region confirms their misrepresentation in the original layout.

The cube structure is clearly visible from Figure 7a, coloured dark to light from top to bottom. In Figure 7b, we show a Shepard diagram of the layout generated by PCA. Each point in the Shepard diagram represents a distance between a pair of objects. In Figure 7b, we have highlighted a section of points in the upper left of the layout: those points corresponding to the relationships worst represented in the PCA layout. Figure 7c shows the how this selection affects the scatterplot display. The linking between views informs us that the objects worst represented in the layout appear in the centre. These objects represent points at opposite corners of the cube, forced together in the projected layout.

Having identified such a poorly represented area of the layout, it may be desirable to extract the subset of objects in that region and lay them out separately. In doing so, we remove the influence of the full data set, and examine only relationships between the objects in that subset. The selected region was therefore fed into another PCA module and re-projected, yielding the layout shown in Figure 7d. It can be seen that the inter-object distances are now more accurate; the two corners of the cube have been separated. As a quantitative measure of the quality of the layouts, the stress of the full layout was measured and compared to that of the sub-layout. As expected the stress of the sub-layout was much less than that of the global layout: 0.001 compared to 0.149.

4.2.2. Antarctic data set. The above scenario demonstrated, via a simple example on synthetic data, how a profiling component could be used interactively in combination with other views to encourage further exploration of a data set. We now provide a similar example to illustrate the usefulness of such techniques in a real-world setting. The data were gathered with a remote sensor probe during an investigation into carbon cycling in Antarctic lakes. They represent a number of properties measured over time, such as water temperature and the level of photosynthetically active radiation.

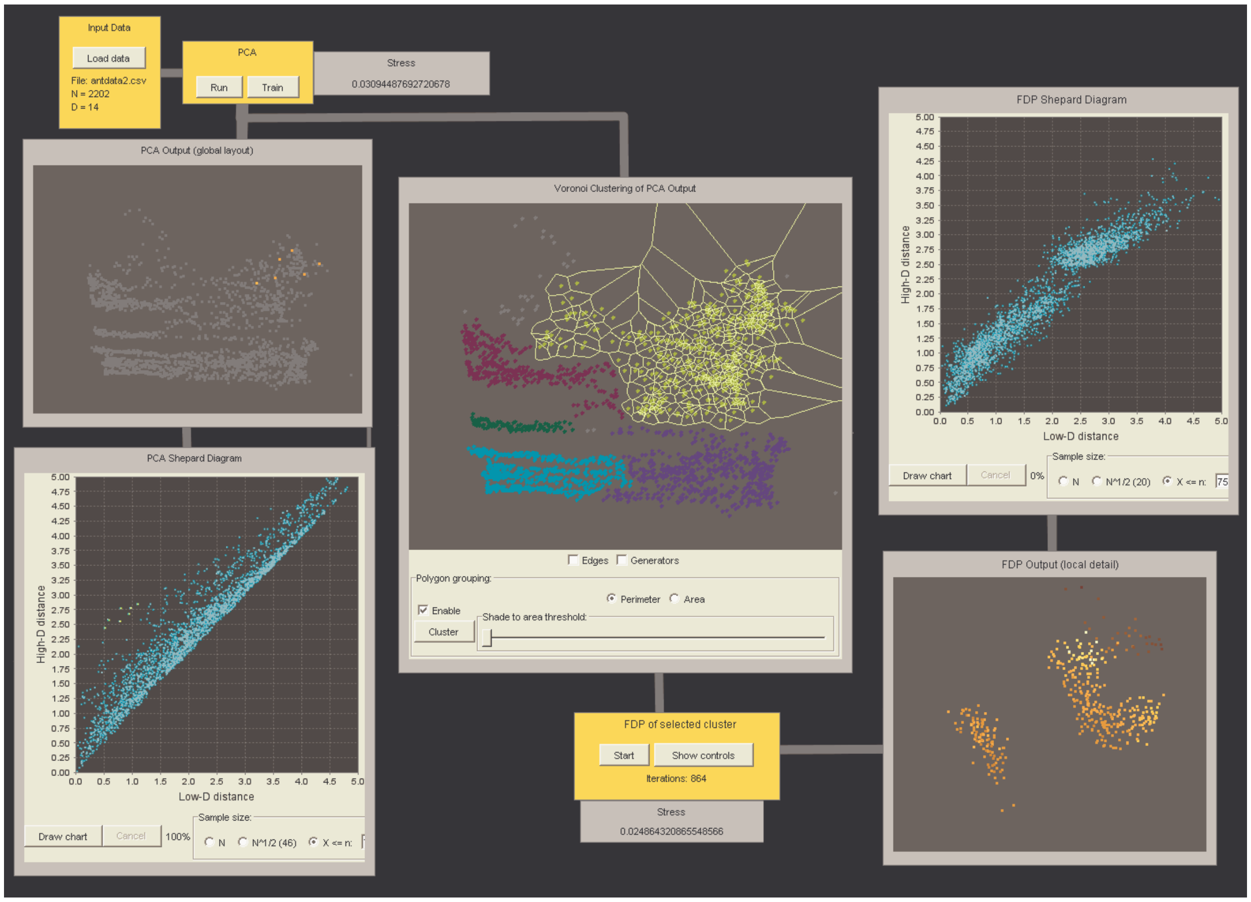
The data set was initially fed into a PCA module, with the Shepard diagram used to identify a local region of items that were potentially badly placed. In a manner similar to the previous example, points far away from the diagonal trend were selected in the Shepard diagram, which resulted in the contributing items being highlighted in the connected PCA layout. The leftmost two components of Figure 8 illustrate the scatterplot and Shepard diagram following this selection.

Having identified these poorly represented objects, we may observe that they appear to be localised to a specific region in the top-right of the layout. We may hypothesise that this area could represent a distinct cluster within the data, which has not been made apparent by the PCA layout. A Voronoi component may be employed to gain a clearer understanding of the partitions within the data. The output from PCA is fed into a Voronoi component, which identifies five separate clusters in the layout.

The Voronoi component is illustrated in the centre of Figure 8. Five clusters were found, and shown in different colours. Outlying objects not identified as belonging to a specific cluster were coloured grey. It may be seen that the objects highlighted in the PCA layout all belong to the yellow cluster. We select this subset and overlay the Voronoi tessellation.

Having now identified a cluster of the data within which certain distances are poorly represented, we may extract it and apply further processing to determine why this is so. Through connecting to the Voronoi output port, another component may take as input the selected cluster. The figure illustrates how we pass the cluster to an FDP routine. This nonlinear technique is able to discover further detail that PCA could not identify: two clear sub-clusters are found within the selected data.

The PCA layout had clearly failed to adequately separate these two sub-clusters, which explains the large discrepancy between high and low dimensional distances observed from the Shepard diagram. The measures derived from the stress component confirm the findings, with the PCA layout giving 0.031 and the FDP layout of the extracted cluster giving 0.025.



File: antdata2.csv
N = 2202
D = 14

Stress: 0.03094487692720678

FDP Shepard Diagram sample size: 0% [✓ X <= n: 75]

PCA Shepard Diagram sample size: 100% [✓ X <= n:]

Polygon grouping: [✓ Perimeter] Area; [✓ Enable]

Iterations: 864

Stress: 0.024864320865548566

Figure 8. Various algorithmic and visualisation components working together in a coordinated environment. A PCA layout is made, and the associated Shepard diagram used to detect a local area that might be better represented if considered separately. A Voronoi tessellation component is used to cluster the data, and extract the cluster containing the previously identified local area (the yellow objects in the central component). This cluster is processed with an FDP routine, which uncovers 2 sub-clusters that we had not previously been able to identify.

Having discovered the presence of two sub-clusters, we may be interested in seeing how they were depicted in the original PCA layout. Comparing the FDP layout and the Voronoi display, it may seem as if the smaller of the two sub-clusters appears on the left of the yellow Voronoi region, with the larger C-shaped sub-cluster appearing on the right. Had the two images been produced independently, one may have made this assumption. HIVE's interactive, coordinated view framework, however, allows users to compare the location of the same objects in different layouts. Figure 9 shows the selected C-shaped sub-cluster in the FDP layout and the resultant highlighting of the corresponding objects in the PCA layout. It may be seen that the division between the two sub-clusters actually occurs in the top-right corner of the PCA layout.

The final stage of processing undertaken in Figure 8 is the creation of the Shepard diagram of the FDP layout, shown in the top right corner. In tandem with the stress calculations, this allows a visual comparison of the degree to which the PCA and FDP layouts preserved high-dimensional relationships.

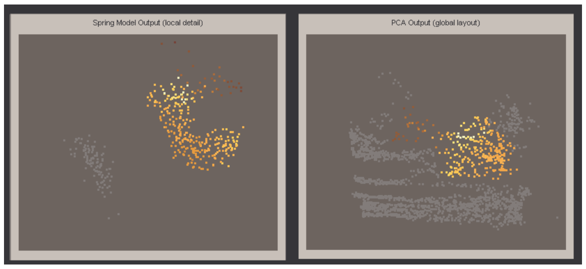

Spring Model Output (local detail)
PCA Output (global layout)

Figure 9. The left image shows the FDP layout of a selected Voronoi cluster within a PCA layout. On the right is the original PCA layout. Selecting the C-shaped sub-cluster on the left highlights the corresponding objects in the PCA layout, helping us to understand the overlap or separation of sub-clusters in the PCA.

The Shepard diagram resulting from the PCA layout exhibits a 'cleaner' line on the 45 degree diagonal, as we would expect from the discussion in section 3.6. It is the FDP's Shepard diagram, however, that appears to show less deviation from the diagonal overall. This is perhaps better illustrated in Figure 10, where the 45 degree diagonal line has been drawn.

It may be noted that two separate clusters of points exist on the Shepard diagram of the FDP layout. Again, an initial reaction may be to assume that each of these corresponds to one of the sub-clusters identified in the FDP layout. This is not the case, although the 'clustering' of the Shepard diagram is due to the presence of two clusters in the FDP layout. The Shepard diagram plots distances, and therefore the two apparent distinct groups of points on the Shepard diagram correspond to distinct ranges of distances within the layout. The lower of the two groups refers to pairwise distances between objects in the same sub-cluster, whereas the higher group represents inter-cluster pairs.

These examples have illustrated the advantages of HIVE's multi-view framework over single static layouts. The provision of profiling modules provides further insight into data sets. Linking and interactive coordination between such views encourage further exploration and leads to greater understanding.

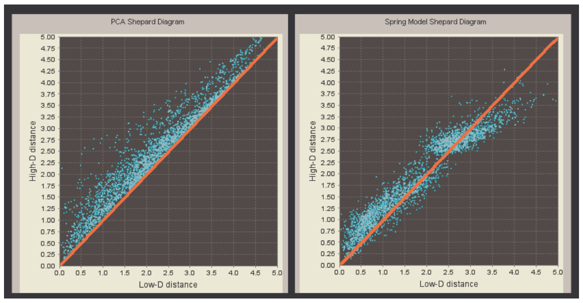


Figure 10. Shepard diagrams based upon the PCA layout of the full data set (left) and the FDP layout of the selected cluster. Red lines are drawn at 45 degrees to help detect the extent to which points deviate from this diagonal.

5. Future work

In this section, we examine some areas of related work that may lead to interesting future avenues of research.

We are currently investigating the incorporation of new analysis tools in HIVE. The force-directed placement algorithms developed in HIVE reduce the dimensionality of the data so that they can be plotted in 2D space. However, we generally rely upon the human visual system to pick out clusters and other interesting patterns; such structure is not formally classified. These perceptual skills are powerful but have limits, and so we are currently working upon improving the Voronoi–based clustering in layouts, and developing ways of determining the validity of identified clusters.

Due to HIVE's architecture individual clusters may be extracted from one view and piped through further transformations and into other views. It would be interesting to assess the homogeneity of these clusters and also ascertain which individual data elements contribute most to layout stress and error.

Functionality originally devised for algorithmic components can also extend to these profiling modules. For example, in [6] it was described how HIVE could automatically generate a recommended hybrid algorithm based upon input data and a user's selection of visualisation components. The generated algorithm is part of a 'cookbook', where each 'recipe' is a set of connected components, selected to provide an efficient solution by considering dimensionality and cardinality of the input data.

While such automated algorithm generation may make it easier for users unfamiliar with the algorithms to build visualisation applications, the purpose of algorithmic parameters would remain unclear. For example, force-directed placement algorithms typically involve constants for properties such as damping and tension, and a module in HIVE may allow user specification of such constants. In such cases, analysis tools could also be automatically generated at appropriate locations to aid the user's understanding of the effects of these parameters upon algorithm performance. This feedback might enhance the user's comprehension of the solutions generated and, in turn, encourage further experimentation and deeper understanding.

Another area that might be of interest is exploring the impact made by algorithmic components upon system memory. It may aid designers to have a module within HIVE that can measure and visualise this.

6. Conclusions

We have presented a number of components within the HIVE system that are used to profile, understand and control other HIVE components. We outlined their construction, and gave examples of their use and utility. One of the advantages of bringing such components into a visualisation system is to support the process of understanding the strengths, weaknesses and interdependencies of algorithmic components. Through techniques such as recording and displaying the performance of ongoing runs, and linking layouts from complementary algorithms and from intermediate stages, we suggest that designers and users can explore not only the data but also the ways that the system represents, transforms and presents that data.

More generally, this work explores the way that making a visualisation that is customised to one's data and interests, and which takes advantage of a palette of algorithmic components, can be a complex task—a task that may be aided by modern tools for interaction and visualisation. It would be frustrating and limiting for designers and for users if powerful tools for analysis and understanding data were themselves difficult to analyse and understand. Therefore we suggest that the use of visualisation for visualisation – in the form of well-designed interaction with the components, processes and parameters of a visualisation system – may afford deeper insight into the visualised information itself.

Acknowledgements

We thank Stefan Rennick Egglestone and Chris Greenhalgh (U. Nottingham) for the Antarctic data.

References

[1] Warren Torgerson. Multidimensional scaling: I. Theory and method. *Psychometrika*, 17, 401-419. 1952.

[2] Roger Shepard. The analysis of proximities: multidimensional scaling with an unknown distance function. *Psychometrika*, 27, 2, 125-140. 1962.

[3] Matthew Chalmers. A Linear Iteration Time Layout Algorithm for Visualising High-Dimensional Data. In *Proceedings of IEEE Visualization* 1996, San Francisco, 127-132. 1996.

[4] Tuevo Kohonen, Kaski, S., Lagus, K., Salojrvi, J., Paatero, V., Saarela, A. Self Organization of a massive document collection. *IEEE Transaction Neural Networks*, 11, 3, 574-585. 2000.

[5] Alistair Morrison, Greg Ross and Matthew Chalmers. Fast Multidimensional Scaling through Sampling, Springs and Interpolation. *Information Visualization 2*, 1. 68-77. 2003.

[6] Greg Ross and Matthew Chalmers. A visual workspace for constructing hybrid multidimensional scaling algorithms and coordinating multiple views. *Information Visualisation 2*, 4. 247-257. 2003.

[7] Peter Eades. A heuristic for graph drawing. *Congressus Numerantium 42*. 1984.

[8] Mu-Chun Su and Hsiao-Te Chang. Fast Self-Organizing Feature Map Algorithm. *IEEE Transactions on Neural Networks*, 11, 3, 721-733. 2000.

[9] Dominique Brodbeck and Luc Girardin. Combining Topological Clustering and Multidimensional Scaling for Visualising Large Data Sets. Unpublished paper (accepted for, but not published in, *Proeedings IEEE Information Visualization 1998*).

[10] Alistair Morrison and Matthew Chalmers. M. Improving hybrid MDS with pivot-based searching. *To appear in Information Visualization 3*, 2. 2004.

[11] Chris North and Ben Shneiderman. Snap-together visualization: can users construct and operate coordinated visualizations? *International Journal of Human-Computer Studies 53*, 715-739. 2000.

[12] Nadia Boukhelifa and Peter Rodgers. A Model and Software System for Coordinated and Multiple Views in Exploratory Visualization. *Information Visualisation 2*, 4. 258-269. 2004.

[13] Craig Upson, Thomas Faulhaber, David Kamens, David Laidlaw, David Schlegel, Jeffrey Vroom, Robert Gurwitz and Andries Van Dam. The application visualization system: a computational environment for scientific visualization. *IEEE Computer Graphics and Applications*. 30-42. 1989.

[14] Paul Haeberli. ConMan: a visual programming language for interactive graphics. *Computer Graphics*, 22, 4, 103-111. 1988.

[15] Ramana Rao and Stuart Card. The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus+Context Visualization for Tabular Information. *In Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems. ACM*, April 1994.

[16] Andreas Buja, Deborah Swayne, Michael Littman and Nathaniel Dean. XGvis: Interactive data visualization with multidimensional scaling. *Under review Journal of Computational and Graphical Statistics*. 1998.

[17] Joseph Kruskal, Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis, *Psychometrika*, 29, 1-27. 1964.

[18] Jonathan Cohen. Drawing graphs to convey proximity: an incremental arrangement method. *ACM Transactions on Human-Computer Interaction 1997*, 4, 3. 197-229. 1997

[19] I Borg, P. J. Groenen, Modern Multidimensional Scaling: Theory and Applications, Springer. 1997.

[20] A. Okabe, B. Boots, K. Sugihara, S.N. Chiu. Spatial Tessellations. *Concepts and Applications of Voronoi Diagrams*. Wiley, Chichester, second edition, 2000.