Visualisation Techniques for Users and Designers of Layout Algorithms

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

Abstract

Visualisation systems consisting of a set of components through which data and interaction commands flow have been explored by a number of researchers. Such hybrid and multistage algorithms can be used to reduce overall computation time, and to provide views of the data that show intermediate results and the outputs of complementary algorithms. In this paper we present work on expanding the range and variety of such components, with two new techniques for analysing and controlling the performance of visualisation processes. While the techniques presented are quite different, they are unified within HIVE: a visualisation system based upon a data-flow model and visual programming. Embodied within this system is a framework for weaving together our visualisation components to better afford insight into data and also deepen understanding of the process of the data's visualisation. We describe the new components and offer short case studies of their application. We demonstrate that both analysts and visualisation designers can benefit from a rich set of components and integrated tools for profiling performance.

Categories and Subject Descriptors (according to ACM CCS): H.5.2 [Information Interfaces and Presentation]: User Interfaces; I.3.6 [Computer Graphics]: Methodology and Techniques – Interaction techniques

1. Introduction

As the field of Information Visualisation has matured, its tools and techniques have grown in number and variety. Driven by application areas that demand interactive exploration of large amounts of data, and exploiting advances in the hardware available for display, processing and storage, new methods of interaction and analysis are continually being added to those we have available. Designers and users are faced with an increasingly large set of alternatives and combinations, and this may be challenging or problematic at times. However, this growing set of tools also opens up possibilities for complementary views of data, and more efficient forms of analysis. Visualisation traditionally concerns the richer interpretation of data that comes from revealing and exploring data patterns, combinations and structures, but there is also a line of work that deals with linking and combining tools and algorithms so as to support such interpretation. Our research aims to contribute to this latter line of work by presenting a set of tools for information visualisation within a framework for combining them.

Our recent work builds on hybrid algorithms such as those described in [10] and [11], which produce 2D scatterplot views of multivariate data, where point proximities match as closely as possible those in the original high-dimensional space. The presented algorithms were hybrid in nature, involving a sequentially applied set of techniques that enabled faster layout of relatively large data sets. Use of computationally expensive algorithms was restricted to a subset of the data to form a basis for comparatively inexpensive interpolation of the rest of the data. This flow of selected subsets of the data through the sequence of algorithmic components also influenced the design of a graphical environment for selecting and combining visualisation components, called HIVE (Hybrid Information Visualisation Environment) [12]. HIVE is targeted at both the user and the designer of layout algorithms and draws upon design approaches such as dataflow architectures [15] and visual programming [7]. This allows the intuitive depiction of algorithmic and display components as nodes in a graph where the arcs represent either the flow of data between components, or interconnection of components to support brushing and linking [1, 2]. It should be noted that visual programming is related to algorithm animation and program visualisation [3, 14], but the distinction between them is that program visualisation is generally applied to help understand algorithms and programs that have been

created in a conventional textual programming language. However, visual programming entails program creation solely by graphical constructs.

In this paper we report two new HIVE components. One component is designed for the data analyst to enrich an existing view and one is for the designer of layout algorithms, offering facilities for evaluation and profiling of visualisation applications built within HIVE. The process of dimension reduction is generally unsupervised and until recently there have been few attempts at providing support for user-intervention [16]. The first of our new components is for the user and enhances an existing view to provide the user with interactive functionality. This work focuses on force-directed placement (FDP) algorithms – a family of techniques that simulate physical forces acting on objects to iteratively produce layouts from an initial random starting configuration [5, 6]. We postulate that a benefit of iterative solutions is the animation of the layout process, which provides feedback on the algorithm's progress. A traditional shortcoming is that the model might occasionally converge to a local minimum – a non-globally optimal configuration. The component presented in this paper provides the user with the ability to interactively control the addition of energy into specific parts of the model to help it break free from local minima. We also show how to interactively identify such minima.

While the first new component helps the user to intervene in the process of generating a layout of data, the second new component is concerned with the retrospective analysis of algorithmic performance. Algorithms for creating layouts of multidimensional data often behave differently given data sets of varying dimensionality and cardinality. These algorithms can be assessed according to the quality of the layout as well as run time. Previous work has shown that hybrid algorithms can be efficient in generating layouts. However, the hybrid combinations are often determined heuristically and therefore the effects of different algorithmic stages on run times and layout quality might not be well known. To address this and provide the algorithm designer with some insight into the effects of different hybrid combinations, a new HIVE component has been developed to collate the results from batch runs of algorithms with different parameter settings and on different data sets. The results of these batch runs are, themselves, multidimensional data and therefore lend themselves to visualisation via the algorithms upon which they report.

We aim to allow the designers and users of visualisation algorithms to use such sophisticated interaction techniques to assess the performance and variability of the algorithms themselves, thus affording better understanding of the overall system and insight into the data being visualised.

In the next section of this paper we discuss the motivation behind the algorithms we have created and describe HIVE, the environment in which they are developed, used and profiled. Section 3 provides details on the interactive technique for addition of energy during the layout process. In Section 4, we go on to illustrate how the algorithm designer can benefit from algorithm profiling. Throughout, we provide brief case studies outlining the use of the extended version of HIVE to analyse data sets and the process of visualisation. We offer some suggestions for future work before concluding the paper with a summary, and the overall direction of our research.

2. Algorithm profiling and HIVE

To gain actionable knowledge from the ever-increasing sea of data facing analysts, data must be represented in such a way that any pertinent information is made available as quickly as possible. It is well known that of all the senses, the human visual system has by far the greatest bandwidth for communicating information to the brain and it is for this reason that data are often represented graphically so as to appeal to the human's visual perception [4].

The challenge, however, is in graphically depicting abstract data. Abstract data are those observations or measurements that have no direct physical derivation and therefore do not immediately lend themselves to the spatial mappings required for visual rendering. This is compounded by complex data sets consisting of many items, each consisting of many variables. In the effort to make sense of these multivariate data by thinking of them in spatial terms, they are referred to as *multidimensional* data.

When data are presented graphically on a spatial substrate, interesting features such as patterns, trends and Gestalt forms might be revealed. A very popular means of achieving this is to plot the data as points against the axes of a two-dimensional scatterplot. If the data dimensionality is relatively low, then scatterplot points can be rendered as glyphs, whose positions denote two dimensions, and whose visual properties encode the remaining dimensions in retinal variables such as shape, size or colour [4]. However, when the data dimensionality is too high to directly map to position and other visual structures, the data must be transformed in such a way that they are represented by a lower number of derived dimensions that retain as much of the original information as possible. This is known as the challenge of dimension reduction.

Many researchers have developed dimension reduction algorithms, sometimes referred to as layout algorithms, each with different benefits and drawbacks. Some algorithms are effective at reducing dimensionality

while preserving the high dimensional relationships but are too inefficient to scale up to large data sets. Conversely, some algorithms might be fast but are unable to accurately capture the original information and therefore fail to display certain interesting patterns. To address this, our previous work has investigated the diligent combination of algorithms to minimise the individual weaknesses while utilising their strong points [10, 11].

HIVE is a system developed for the user to visualise data and for the designer to develop and profile visualisation algorithms. It is a toolkit of algorithmic components that can be used either individually, or integrated into hybrid algorithmic models. As well as being a data exploration environment, HIVE is also a useful system within which to profile and evaluate hybrid algorithms. Several novel modules have been implemented to measure and display performance characteristics of other HIVE modules. Such profiling modules permit algorithm evaluation to be interactively coupled with the algorithms being run; visual metaphors used for the interconnection of algorithmic components can also be used in linking together profiling tools. The decision as to which properties of algorithmic performance to measure can be made and altered at run-time.

Various modules have been developed, for simple measurements such as run time and metrics of layout quality, and for more complex analysis and profiling. Several of these components are shown in Figure 1, which depicts an FDP routine being used to create a 2D scatterplot layout (top right) of a synthetic data set representing a 3D cube. A tool is added to measure *stress* – a metric measuring the residual sum of squared error between high dimensional and layout distances [9]. Stress is measured after each iteration of the FDP and graphed in a chart component, which can be used in conjunction with the scatterplot to determine the extent to which the FDP is approaching convergence. A Shepard diagram [13] in the bottom right may also be used in evaluating the quality of layout obtained. Each point in this diagram corresponds to a pair of objects, plotting their high dimensional distance against their layout distance. Functionality is provided to support brushing and linking between the Shepard diagram and scatterplot layout. A user can therefore select a distance in the Shepard diagram, and be shown the corresponding objects in the scatterplot. Distances far from the diagonal can alert a user to areas of the layout (perhaps regions of local minima) that may benefit from further processing.

Data are loaded from the multiple runs module in the top left of the figure. This module can coordinate multiple executions of algorithms, taking a series of commands from the user and passing the appropriate parameters for each run to various algorithmic stages. Used with the chart component, this allows models to be compared over a variety of data sets and under various algorithmic conditions, with results being automatically charted as part of an unsupervised process.

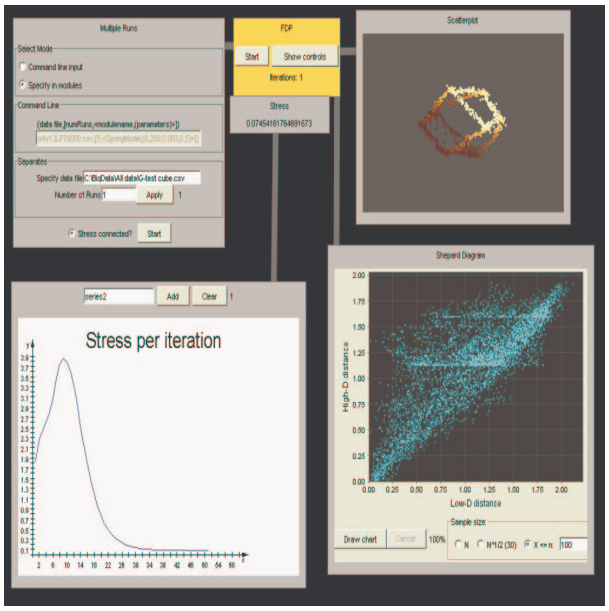

Figure 1: An example of several profiling modules in HIVE, applied to an FDP routine running on a synthetic 3D cube. The FDP produces 2D coordinates, used to plot the layout in the top right image. A stress module is connected to the FDP output, and feeds into the chart in the bottom left. Thereby, stress may be measured and plotted after each iteration. Finally, the Shepard diagram in the bottom right plots high dimensional distances against low dimensional (layout) distances.

3. Interactive addition of energy

In Section 2 a force-directed placement routine is used to produce a layout. FDP is an iterative process in which objects are initially placed at random and discrepancies between inter-point distances in the 2D layout and the high-D (variable) space are modelled as forces. Take for example two points that are much closer together in the layout than in the high-D space; these two points will experience a repulsive force and the algorithm will try to push them further apart.

Iterative layout techniques can sometimes find locally optimal configurations, but not necessarily globally optimal ones. A simple example of this is a layout becoming 'twisted', as illustrated in Figure 2, showing a 2-dimensional data set representing a rectangle. Objects at either side of the twist are locally

well positioned, but the global structure of the layout is flawed.

Iterative models can often recover from such minima. An example such as the one illustrated in Figure 2 would almost certainly reach a globally optimal state eventually. As the section below the fold has fewer objects than the one on top, it is this bottom section that is more likely to conform to the top half's orientation. This recovery process, however, can be very time consuming. Since the objects in the lower section are well positioned in their local neighbourhoods, and reasonably well positioned in regard to the distant objects, there will be only very small composite forces pulling them in the correct direction.

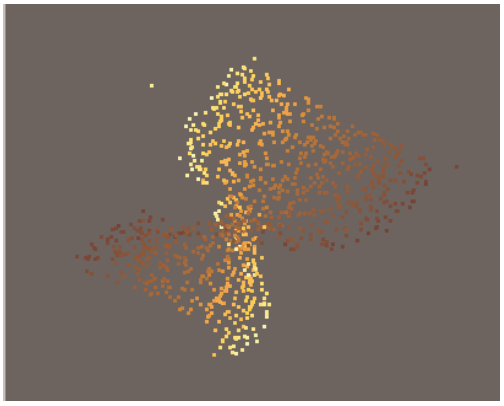

Figure 2: An FDP routine executing on a set of rectangular data has found a local minimum; the layout has become twisted. Since relationships are well represented within regions at either side of the folded area and the 'errant objects' are roughly well-positioned in regard to distant objects, composite force vectors acting on them will be small. As such, the layout may take a long time to unfold.

Figure 3 shows graphs of stress per iteration for two executions of an iterative FDP model on the data set illustrated in Figure 2. Both graphs have the general shape associated with such models: stress rises for the first few iterations as large forces are generated between objects in initial random positions. Thereafter, stress decreases as the layout reaches its conclusion. The first case shows an average run, where the model proceeds steadily towards its final configuration. As the data set is two-dimensional, the stress of the final layout should reach zero, and this is achieved after 52 iterations. The second graph in Figure 3 illustrates a case where the layout has developed a twist as in Figure 2. This graph shows a period of very slow stress decrease of around 200 iterations, before it finally moves towards zero stress after 280 iterations. This 'plateau' represents the period where the fold has occurred, and the system slowly becomes untangled. Although the layout is clearly flawed, stress is relatively low during this period, as most of the inter-object relationships are well represented.

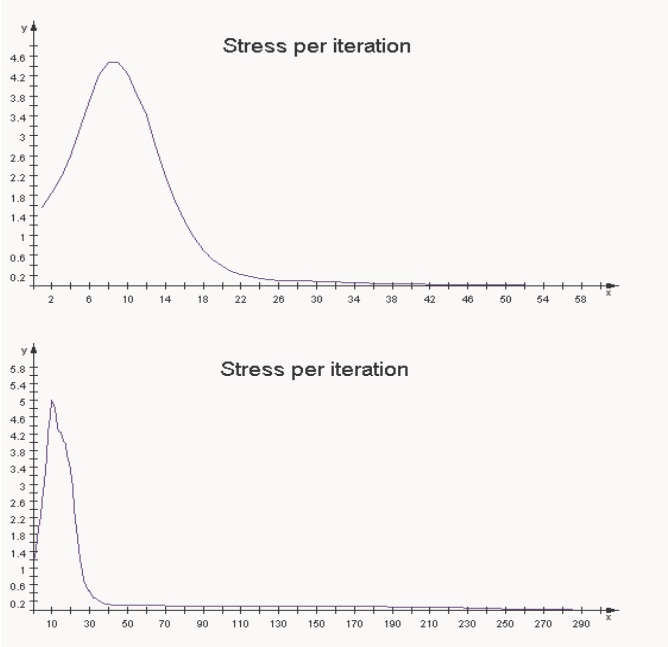

Figure 3: Two graphs of stress over iterations for FDP executing on the 2D rectangular data. With 2D data, a layout with stress of zero is possible. In the first case, the algorithm proceeds without a hitch and completes after 52 iterations. The second chart, with a different horizontal scale, shows a long 'plateau' starting at 40 iterations as the layout becomes twisted. It eventually reaches zero stress after 280 iterations.

A user observing layout formation may find this very frustrating. It may be readily apparent to an expert user that such a fold has occurred, but there is no way of interacting with the process to untangle the layout. The user therefore has the option of watching the layout attempt to unfold, or abandoning execution and restarting.

We propose novel functionality to allow interaction with the layout in such situations, to help accelerate the untangling process. Through interacting with the scatterplot window, we allow users to add energy to localised regions of the system to increase the forces acting on the misplaced objects. The additional energy increases the magnitude of forces acting upon each object, encouraging wayward regions of the layout to reach their destinations more rapidly. Conceptually, this is similar to throwing a pebble into a pond. Energy is

added to a specific location, and slowly dissipates with distance from the point of 'impact'.

Controls are provided to allow the user to specify the location at which to add energy, as well as the amount of energy and the range over which it should act. The user double clicks on the target area of the scatterplot. A red circle is drawn, representing the range, which steadily grows until the mouse button is released. The user may first zoom in, in order to gain fine-grain control of the layout area—and therefore the objects—to which the energy should be applied. Conversely, by zooming out, a large area of the layout may be targeted quickly. The amount of energy to add may be specified via a slider at the bottom of the scatterplot component. These controls are illustrated in Figure 4.

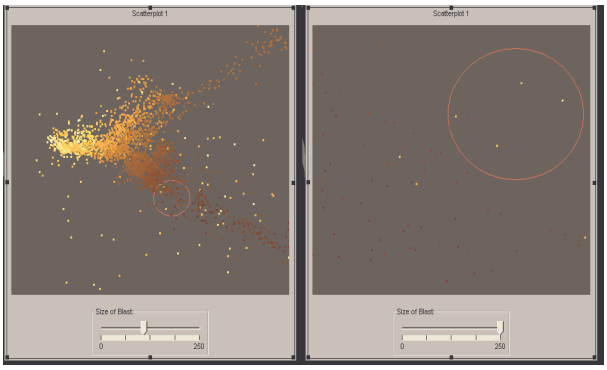

Figure 4: An illustration of the controls provided to add energy to a layout. Double clicking on the layout draws a circle, which steadily increases in size until mouse release, allowing specification of the area to which energy should be added. A slider at the bottom of the panel controls the amount of energy applied. By zooming in, as in the image on the right, finer grain control may be exerted.

On mouse release, the desired amount of energy is added to the layout at the specified location. Energy is at its peak at the original location of the user's double click and reduces with distance, reaching zero on the edge of the circle.

Figure 5 illustrates the usefulness of this functionality. A user may detect that a fold has occurred in an FDP layout (after 70 iterations in this example). It was illustrated in Figure 3 how many iterations it may take to resolve such an occurrence. The user therefore adds energy to a relatively local area within the fold. The energy added to these points scatters them, leaving a hole in the layout after 75 iterations. Since these objects are relatively few, however, this does not make a great impact on the general layout structure, and the wayward objects are soon pulled back into position. Hereafter, the objects will move towards their ideal locations faster, as the composite forces acting upon them will be greater. The layout has been allowed to 'bounce out' of its local minimum. After 85 iterations, the fold has almost been resolved, and after 95 the layout is near completion.

This process is similar to simulated annealing [8], which is an automated process based upon a metaphor of the movement of atoms in a cooling metal. The distinction here, however, is that the user is at the centre of the operation. Energy is added at the user's discretion, rather than in accordance with a cooling schedule. Another distinction is that energy is added to localised regions, rather than globally, so settled areas of a layout can be left unperturbed.

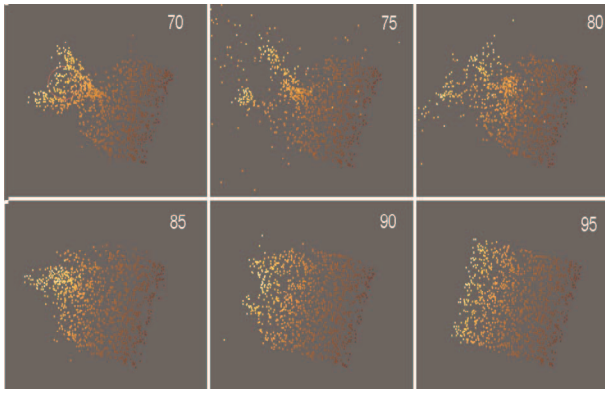

Figure 5: A twisted layout shown at 5 iteration intervals (specified top right in each frame). Energy is added to the twisted region after 70 iterations, allowing the rapid convergence of the model.

While the ability to help the layout bounce out of local minima is clearly beneficial, it is expected that detecting the minima by visual examination of the layout will be a challenge even for expert users. We have implemented an interactive Shepard diagram to help in this regard. As described in Section 2, the Shepard diagram plots all, or a sample of, pair-wise high- dimensional distances against their representative low-dimensional distances. In an ideal layout, the low-D distances will be equal to the high-D distances and the Shepard plot would depict a straight 45-degree line of positive slope. However, in the typical case with real data, it is impossible to gain such a layout and as a result the discrepancies between the true distances and the layout distances appear as points in the Shepard diagram that deviate from the 45-degree slope.

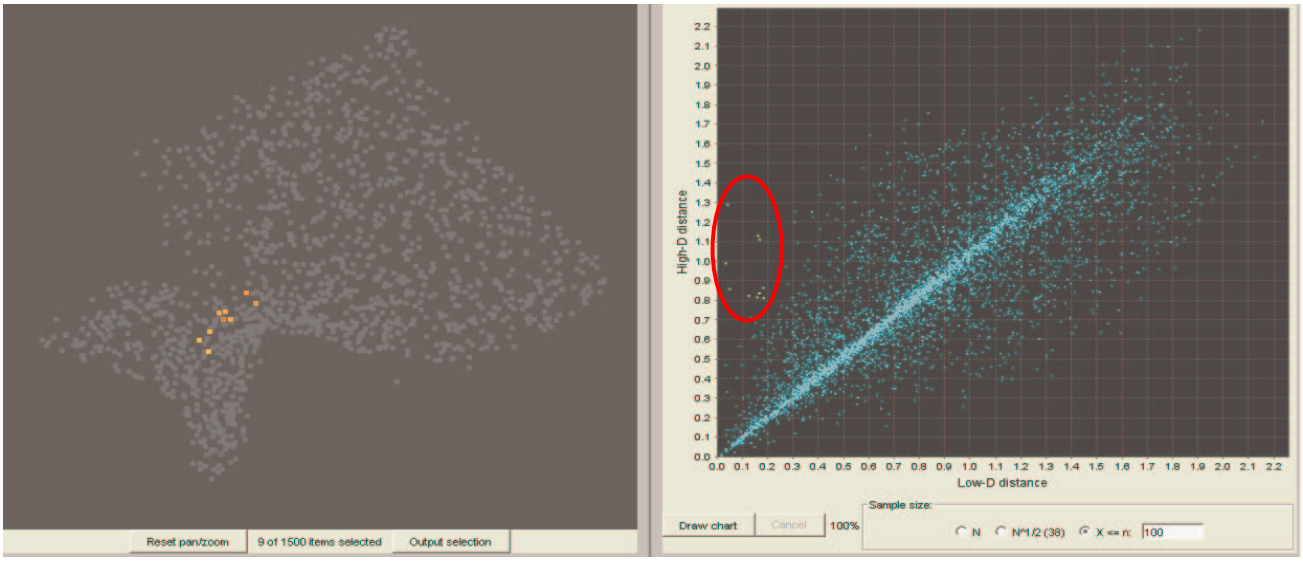

Figure 6: A twist has become apparent in the 2D layout as shown in the left image. This is reflected by a number of points deviating from the diagonal of the Shepard plot on the left. Points that lie above the diagonal represent points in the layout that are too close together. Upon selecting some of these points (highlighted in yellow and circled for clarity) in the Shepard plot, the corresponding points are highlighted in the layout providing guidance as to where to interactively add energy as described earlier.

The Shepard diagram is traditionally a static presentation; the user is afforded no interaction with it. However, it has been realised that by supplementing it with brush and link functionality, we can link it to a layout of high-dimensional data to identify the points that contribute to local minima. Figure 6 shows how a Shepard plot has been linked to a layout similar to that shown in Figure 2. It can be clearly seen that there are some points deviating from the 45-degree slope of the Shepard diagram. Some of these points have been selected in the diagram and as a result, the corresponding points in the layout are highlighted. This tells us where to add energy to the layout to help dispel the local minimum and allow the layout to converge at a faster pace.

The force-directed placement algorithm is based upon the intuitive analogy of forces converging to equilibrium in a mechanical system. In compliance with this simple model, this new method of interactively adding energy to the system provides an intuitive means of user intervention to free it from local minima.

4. Collation of algorithmic profiling results

A new component has been designed to collate and visualise the results of numerous profiling tools descried in Section 2, drawing them together into one unit. Figure 7 illustrates a simple case, where a Multiple Runs module controls several executions of an FDP routine. The algorithm is run on data sets ranging in size between 3000 and 9000 objects, and for varying amounts of iterations. The collation module receives values for the data set size, number of iterations performed, stress and run time. These are then displayed in a table component, with each row corresponding to one algorithmic run.

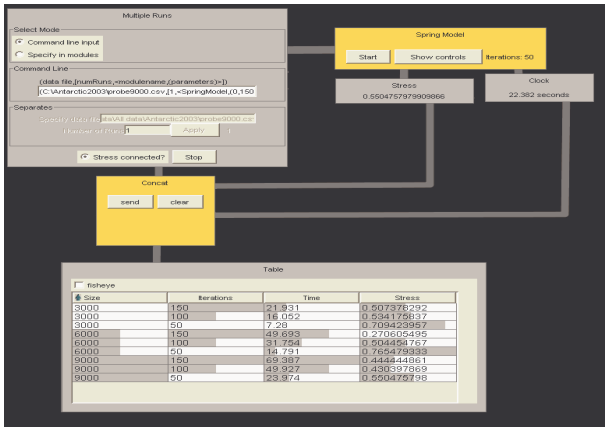

Figure 7: An illustration of the process by which results of algorithmic profiling may be collated. Several runs of a FDP model are performed, with the results of each collected in the collation component and displayed in a table component. Each row represents one execution of the FDP model.

It could be seen from this example that the collation module could gather a number of values into a multidimensional data set, and that the output from the module could therefore be treated in a manner similar to a data source component; it could be used as input to visualisation algorithms. It would be possible, therefore, to create a layout of results, within which to search for patterns. This brings about the possibility of evaluating a set of visualisation techniques, using these same techniques to explore the results. This is achievable within the same interaction framework, and may be performed at run time.

The following example (Figure 8) illustrates this concept. A set of algorithmic modules have been connected together. The configuration represents the algorithm presented in [10]. The evaluation section of that paper described the comparison of a novel algorithm with two competitor techniques. Details of the algorithms are unimportant for this example; suffice to say that they were hybrid models, each composed of several stages. Run times and stress are measured at various stages, and these results are passed to a collation module, along with information on data size and algorithmic parameters. Six runs are performed with each set of conditions.

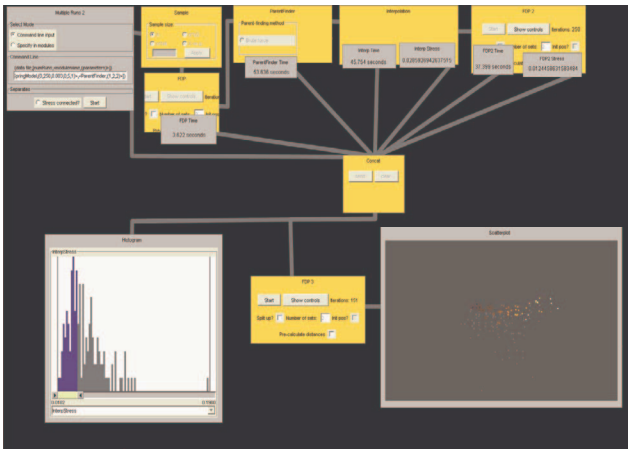

Figure 8: The data flow carries performance characteristics of several stages of a hybrid layout algorithm. The collation module gathers the information and passes it to histogram and FDP modules.

Having gathered the data from the various profiling modules, the collation module feeds into an FDP model to analyse the results. The scatterplot in the lower right corner (and reproduced for clarity in Figure 9), then, represents each algorithmic test run as a visualised object. This may be used in combination with the histogram in the lower left to explore the profiling results.

The interactive controls at the bottom of the histogram component allow each dimension of the data to be examined in turn. Which input dimension to graph may be selected from a drop-down list (in the figure, post-interpolation stress is selected). The double-ended slider may then be used to search over the range of this dimension, with the scatterplot view updating to highlight only the objects fulfilling that range criterion. In the figure, we see that the objects with low values for stress appear towards the top of the scatterplot.

Applying this process to each input in turn, we ascertain that the generated layout is essentially two dimensional. High correlations exist between the data size and run times for the various stages, and these values correspond roughly to the horizontal placement in the layout. The orthogonal dimension corresponds to layout stress, as illustrated in Figure 8 with the histogram.

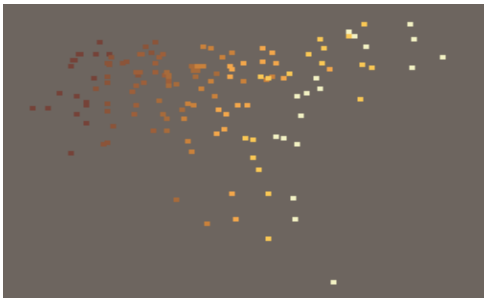

Figure 9: The layout from Figure 8 reproduced for clarity.

One exception to this pattern is the time taken for the final stage of the algorithm. It was noted following interaction with the histogram component that this input dimension corresponded to the vertical layout axis. The final algorithmic stage is a force directed placement routine. This stage executes for as long as necessary to reduce stress to a specific threshold, and therefore its run time is more dependent on the stress level at completion of the previous stage.

The layout is coloured over data set size. Eight different sizes of data were used in the experiment, so each data point is one of eight distinct colours, the brighter shades representing larger sets. It may be seen that the objects representing the smaller data sets tend to appear in lines, forming bands of colour. As the data size increases, however, this effect becomes less pronounced. It may also be seen that the layout appears to spread out as data size increases. This may be explained by the fact that the smaller sets have very similar run times for the various components. As the

data size increases, and run times become longer, the run times exhibit greater variance.

For clarity, this example used a separate FDP module to explore results. However, it is worthy of note that the collated data could have been fed back into the original components i.e. the algorithm that produced the data could itself be used to visualise the results.

5. Future Work

The paper has described new components for data exploration and algorithmic profiling – tools for both visualisation users and designers. We are currently investigating ways of using such tools to automatically detect when iterative layout routines fall into local minima. Stress levels and charting components can already be used to see when such circumstances arise, and HIVE affords brushing and linking between layouts and certain profiling tools to support users in detecting localised areas that make a large contribution to layout stress. However, it would be of benefit to support unsupervised detection of such conditions, thereby allowing HIVE to automatically attempt to resolve certain areas, or extract them for further processing.

We will also continue to examine our novel approach to allowing layout algorithms to reflect upon themselves, as highlighted in Section 4. An algorithm can detect patterns in performance characteristics, with respect to different input data sets and under different algorithmic conditions. This therefore suggests the possibility of using this analysis to automatically modify algorithmic parameters to improve performance.

6. Conclusions

We have presented two components that enrich the set of tools available in HIVE for analysing data and for analysing the visualisation process. We have demonstrated a new tool for allowing the visualisation user to intervene in previously unsupervised FDP algorithms. We have also described a tool for allowing the algorithm designer to profile solutions.

The challenges presented by the hybrid algorithmic approach to dimension reduction include determining which algorithmic components should be combined and in what order, as well as how to assess their performance. The advantages of the combination of algorithms, and the challenges they present motivate us in pursuing this rich avenue of research.

Visualisation and data analysis is often a complex task and, as such, is itself a potential application area for InfoVis tools. Our research develops and explores responses to this, applying analysis techniques to the components, processes and parameters of a visualisation system. This, we suggest, is a promising way to afford better understanding and control of the visualisation system and, in turn, deepen insight into the visualised information itself.

References

[1] Becker R., Cleveland W.: Brushing scatterplots. *Technometrics*, 29, 2 (1987), 127–142.

[2] Boukhelifa N., Rodgers P.: A model and software system for coordinated and multiple views in exploratory visualization. *Information Visualisation*, 2, 4 (2004), 258–269.

[3] Brown, M. H. Zeus: A system for algorithm animation. In *Proceedings of IEEE 1991 Workshop on Visual Languages*, (1991), 4–9.

[4] Card S. K., Mackinlay J. D., Shneiderman B.: *Information Visualisation - using vision to think*. Morgan Kaufmann, 1999.

[5] Chalmers M.: A linear iteration time layout algorithm for visualising high-dimensional data. In *Proceedings of IEEE Visualization 1996*, San Francisco (1996), 127–132.

[6] Eades P.: A heuristic for graph drawing. *Congressus Numerantium*, 42 (1984), 149–160.

[7] Haeberli P.: ConMan: a visual programming language for interactive graphics. *Computer Graphics*, 22, 4, (1988), 103–111.

[8] Kirkpatrick S., Gelatt C. D., Jr., Vecchi M. P.: Optimization by simulated annealing. S*cience*, 4598 (1983), 671–680.

[9] Kruskal J.: Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29 (1964), 1–27.

[10] Morrison A., Chalmers M.: Improving hybrid MDS with pivot-based searching. *Information Visualization*, 3, 2 (2004), 109–122.

[11] Morrison A., Ross G., Chalmers M.: Fast multidimensional scaling through sampling, springs and interpolation. *Information Visualization*, *2*, 1 (2003), 68–77.

[12] Ross G., Chalmers M.: A visual workspace for constructing hybrid multidimensional scaling algorithms and coordinating multiple views. *Information Visualization*, 2, 4 (2003), 247-257.

[13] Shepard R.: The analysis of proximities: multidimensional scaling with an unknown distance function. *Psychometrika*, 27, *2* (1962), 125–140.

[14] Stasko, J. T. TANGO: A Framework and System for Algorithm Animation. IEEE Comp., 23:27--39, 1990.

[15] Upson C., Faulhaber T., Kamens D., Laidlaw D., Schlegel D., Vroom J., Gurwitz R., Van Dam A.: The application visualization system: a computational environment for scientific visualization. *IEEE Computer Graphics and Applications*. (1989), 30–42.

[16] Williams M., Munzner T.: Steerable, progressive multidimensional scaling. *IEEE Symposium on Information Visualization (INFOVIS'04)*, (2004), 57–64.