Info Vis

Regular Article

Information Visualization
12(3-4) 324–357

DOI: 10.1177/1473871612455749
ivi.sagepub.com
SAGE

A survey of two-dimensional graph layout techniques for information visualisation

Helen Gibson, Joe Faith and Paul Vickers

Abstract

Many algorithms for graph layout have been devised over the last 30 years spanning both the graph drawing and information visualisation communities. This article first reviews the advances made in the field of graph drawing that have then often been applied by the information visualisation community. There then follows a discussion of a range of techniques developed specifically for graph visualisations. Graph drawing algorithms are categorised into the following approaches: force-directed layouts, the use of dimension reduction in graph layout and computational improvements including multi-level techniques. Methods developed specifically for graph visualisation often make use of node-attributes and are categorised based on whether the attributes are used to introduce constraints to the layout, provide a clustered view or define an explicit representation in two-dimensional space. The similarities and distinctions between these techniques are examined and the aim is to provide a detailed assessment of currently available graph layout techniques, specifically how they can be used by visualisation practitioners, and to motivate further research in the area.

Keywords

Graph and network visualisation, network layout visualisation, graph layout, force-directed layout, multi-attribute visualisation, 2D

Introduction

A graph can be defined as a set of nodes and a set of edges such that an edge describes the existence of a relationship between two nodes. Drawing a graph can help make better sense of the structure of those relationships than simply looking at the data in tabular form. Simply drawing the graph is not enough as how the graph is drawn has a significant impact on how the graph is understood. Owing to the gestalt principle of proximity, developers of layout algorithms should be aware that nodes placed close to one another will be interpreted by the user as a true relationship whether or not this relationship exists.[1] This means the layout and the arrangement of the nodes strongly influences how the user perceives the relationships in the graph. Therefore, finding a layout which can emphasise relationships, and which does so without misleading the user, is crucial even if further interaction, filtering and analysis may be necessary to discover why those relationships exist.

Even though it is nearly 50 years since Tutte[2,3] proposed his barycenter method (see 'Force-directed layouts'), how best to lay out a graph remains a current problem, one that is still attracting attention. In fact, Blythe et al.[4] asserted that there is no best way to draw a graph and that layout simply depends on which

School of Computing, Engineering & Information Sciences, Northumbria University, Newcastle upon Tyne, UK

Corresponding author:
Helen Gibson, School of Computing, Engineering & Information Sciences, Northumbria University, Pandon Building, Newcastle upon Tyne, NE2 1XE, UK.
Email: helen.gibson@northumbria.ac.uk

features of the graph we wish to highlight. These may be certain aspects of the structure of the graph itself, particular measures of centrality or prominence, or important attributes of the nodes or edges.

Graph drawing has evolved in two different directions: the heavily algorithmic side drawing from mathematical graph theory and the more interactive and application-focused side from information visualisation, often termed network or graph visualisation.[5]

Graph drawing has accumulated a large body of research and its own symposium, 'Graph Drawing',[6] held annually for the past 20 years with articles continuously appearing in the *Journal of Graph Algorithms and Applications* (JGAA). Published in 1999, and building on their 1994 annotated bibliography, Di Battista et al.'s book on graph drawing[7] is regarded as the key reference for an introduction to the algorithmic approach to graph drawing.

This review furthers their research by focusing on the use of graphs in information visualisation through the inclusion of computational improvements to previous methods and, by exploring what makes the key difference between graph drawing and network visualisation, the use of attributes for layout.[8] The work is restricted to two-dimensional (2D) graph layout because techniques that are suitable for two dimensions are not always generalisable to three dimensions; for example, edge crossing is less of a problem in three dimensions than two dimensions,[9] but many 2D layouts consider reducing edge crossings to be of central importance. Other problems include node occlusion, difficulties in finding the best 'view' in space and difficulty in adapting to actions and interactions in three-dimensional (3D) space.[9,10]

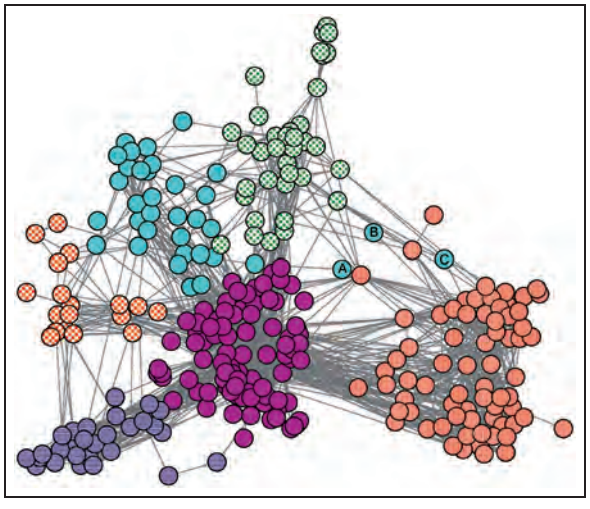

Figure 1. The three labelled nodes (A, B, C) appear separated in the layout from other nodes detected as being in their community.

Di Battista et al.[7] credit Knuth[11] with the first application for automatic graph layout. Knuth's algorithm visualised flowcharts with the aim of improving computer program documentation, and since then network visualisation has been applied to many application areas. These include technological areas such as the structure of the Internet and the hyperlink structure of the web as well as social networks, bibliographic networks and biological networks.

Visualisation of networks should not just be for the sake of it; it should aid the analysis and understanding of the graph. In areas of network analysis, and particularly social network analysis, interpretation and understanding of a graph's structure can come from the calculation of metrics associated with each node of the graph and the graph as a whole,[12] but the idea of drawing a graph is to represent the structure of a graph visually. Reasons to do this are given by Bezerianos et al.,[13] who suggest that visualisation can help with 'detecting, understanding and identifying unexpected patterns' in social networks and, in fact, this could be applied to all graphs. Henry and Fekete[8] go even further by suggesting that 'computing a layout of a graph is necessary to find insights'. This is why visualisation can be useful: it can allow users to see relationships, such as patterns and outliers,[14] that would not be apparent through a metrics-based analysis alone. For example, a metrics-based analysis can tell the user which are the most connected nodes or detect communities whereas a visualisation allows the user to see whether the most connected nodes influence different areas of the graph or if there are outliers to a clustering (see Figure 1). Graph layout algorithms serve to represent a graph subject to some rules or guidelines that should enable this more effective analysis through visualisation.

In information visualisation particularly, drawing the graph can enable both hypothesis generation and confirmation from the data. Much of the research into what can be learnt from studying the properties of a graph concentrates on either network analysis statistics or the study of the topological, static display of the network.[13] Layout that can also stem from interaction with the network, integration of node-attributes (additional data we know about the node) and node metrics (computed statistics that measure properties of the node) encourages further exploration and understanding of the network. The more information we have about the graph the greater the emphasis on good layout algorithms to convey the information in the graph in a way which is informative, accessible and comprehensible to instigate interaction and engagement with the graph by the user.

In this survey 'Family of force-directed and related graph drawing algorithms' covers the major graph

drawing techniques that have been devised and used over the past 30 years; these are divided into three categories: force-directed, those based on dimension-reduction and computational improvements such as multi-level techniques. In the 'Using node-attributes for layout' section techniques developed for using node-attributes more specific to graph visualisation are presented and discussed. Conclusions and scope for further research are covered in 'Discussion'.

Family of force-directed and related graph drawing algorithms

Most algorithmic graph drawing is based on the force-directed paradigm of modelling a graph as a physical system where nodes are attracted and repelled according to some force. The various forms of force-directed graph drawing are among the most frequently used and modified, meaning that they can be be applied to graphs with many thousands of nodes. Although force-directed and dimension reduction-based algorithms are dimension independent, they are most commonly used to produce layouts in two dimensions. The presentation and evaluation of these layouts in this paper are given in a 2D context.

The original algorithm for force-directed graph drawing and its multiple variations are explained in 'Force-directed layouts'. The use of dimension reduction for graph layout shares many similarities with force-directed methods and these similarities and some extensions are discussed in 'Dimension reduction for layout'. More recently, it has become necessary to visualise much larger graphs; thus, many suggestions, including multi-level methods, have been proposed as computational improvements to the layouts in the 'Force-directed layouts' and 'Dimension reduction for layout' sections, along with some spectral methods, which are mentioned in 'Computational improvements'. There are a number of criteria that are used to evaluate the effectiveness of a graph layout. These include computational complexity or running time for the algorithm to execute, the size of graph for which they are realistically able to produce a layout, their ability to comply with certain layout principles or aesthetics, the user's visual assessment and other potentially desirable visual features such as clustering. 'Evaluation of algorithmic layouts for graph visualisation' discusses how the algorithms compare in terms of these criteria and Table 1 gives a summary of this discussion.

Force-directed layouts

Force-directed algorithms were amongst the first to be developed for automatic graph layout and are some of the most commonly used methods today. Based on a physical model of attraction and repulsion, the aim is to lay out the graph optimally. This optimality agreed with criteria put forward as graph drawing aesthetics, as discussed in 'Graph drawing aesthetics'. A forerunner to force-directed methods was Tutte's[2,3] barycentre method. The barycentre between two objects is the point at which the gravitational forces exerted by those two objects cancel each other out. The method was developed for drawing tri-connected and planar graphs in which at least three nodes had a fixed initial position on the external face of a polygon. The unfixed nodes are placed at the barycentre of their neighbours with optimisation through Newton–Raphson iteration. Quinn and Breuer[15] also proposed a force method for placing components on a printed circuit board. It was Eades'[16] spring-embedded force-directed layout that became accepted and inspired many other layout algorithms. Force-directed techniques remain so popular because, in their simplest form, they are not difficult to understand and are easily implemented in code.[7]

Graph drawing aesthetics. Classic force-directed placement approaches apply a set of rules that are said to produce aesthetic graphs. In this context, aesthetics are associated with improving the readability (causing Dunne and Shneiderman[17] to rename them readability metrics) of the drawing[18] on the premise that if the layout makes the relationships in the graph more readable then it is more comprehensible to the user. Good aesthetic properties should ensure that a graph is displayed more effectively and allow the user to easily perceive the topological structure of a graph.[19] This is different from analysing the structure computationally. For example, a user may determine that the connectivity of the nodes follow a power-law distribution or form a small-world network, but visualisation can give greater insight into their structure,[20] such as seeing which clusters in a small-world network are linked or placed close to one another in the layout. Table 2 shows the aesthetic criteria most commonly used in force-directed graph layout with explanations as to why they are considered to produce a good graph layout. However, these criteria are considered intuitive[22] rather than being derived from experimental data.

Incorporating all of these aesthetics into one optimal layout is infeasible because of their competitive nature, as achieving one often requires breaking another.[7] *Additionally*, there is a distinction between what is shown to be computationally aesthetic and what is subjectively 'aesthetically pleasing' to each user;[16] in order to distinguish between these two meanings, from now on the aesthetic graph drawing criteria will be termed principles of graph drawing and aesthetically pleasing will refer to the user's feeling

Table 1. A summary of how the graph drawing algorithms in this article compare. A checkmark against a graph drawing principle indicates that it was considered in the design of the algorithm.

Algorithm	Performance	Graph drawing principles				Size	Notes
		Edge crossings	Symmetry	Even node distribution	Uniform edge		
Spring Embedder	Good for graphs of up to 50 nodes		✓Achieves this for small graphs		✓Small graphs have even edge lengths	50 nodes and should not be dense	Best for fewer than 30 nodes
Fruchterman and Reingold (FR)	Aim was for speed and simplicity but slower than GEM	✓Performs worse than Tunkelang's algorithm on edge crossings	✓Shows symmetry	✓	✓This was one of their aims but they did not think they achieved it; but it is better than SA and Tu for dense graphs	Able to draw graphs with thousands of nodes but slowly, and does not often result in a good layout	Does not require parameters to be optimised
GEM	Much faster than FR and KK	✓Although they say they did not explicitly minimise this	✓Shows symmetry well	✓Similar performance to FR and KK	✓Similar to FR and KK	Handles large (> 128 nodes) better than FR and KK	Produces similar results to FR and KK when compared using the graph drawing principles
Kamada and Kawai (KK)	Much slower than GEM for graphs of > 30 nodes		✓Main aim and is shown in layouts			Up to 30 nodes, better results than GEM on small sparse graphs for graph drawing principles	Euclidean distance should approximate graph theoretic distance
Simulated Annealing (SA)	Considered to be too slow to be used practically	✓Performs worse than FR and Tu	Can show symmetry	✓Performs worse than FR and Tu	✓Performs worse than Fr and Tu	Can be used for graphs of up to 60 nodes with similar performance to FR, KK and GEM	Also tries to prevent nodes coming too close to edges. Adaptable
Tunkelang (Tu)	Similar to FR; 100 nodes in less than 3 s	✓Performs better than FR and SA on this metric	Does not show symmetry at all	For dense graphs FR performs better	✓For dense graphs FR performs better	Tested on graphs up to 60 nodes, sparse and dense. Results are better on the sparser graphs	Aims to place non-adjacent nodes further from each other. If other force-directed layouts do not produce a good result then try this algorithm.
LinLog				Goes against this criterion to show clustering	Violates uniform edge lengths in order to emphasise clustering	Able to draw graphs with many thousands of nodes	Results in graphs with more obvious clusters than KK and FR
Force Atlas	Speed can be sacrificed for greater precision or vice versa. In terms of number of iterations required to lay out the graph it takes fewer than FR			Also tries to optimise clustering but to a lesser extent than LinLog	Edges should be as short as possible	Tested on graphs with more than 20,000 nodes	

(continued)

Table 1. (continued)

Algorithm	Performance	Graph drawing principles				Size	Notes
		Edge crossings	Symmetry	Even node distribution	Uniform edge		
Pivot MDS	Can lay out a graph with 100,000 nodes in 11 s			Approximation to a classic MDS technique which reproduces graph theoretic distances as Euclidean distances. Node distribution is proportional the graph theoretic distribution		More than 100,000 nodes	
HDE	Should be extremely fast; 1 million nodes in less than a minute	Often produces layout with many edge crossings		Similar to Pivot MDS and places adjacent nodes close together and non-adjacent further apart	Large variation in edge lengths in larger and more challenging graphs	More than 1 million nodes	Static layouts can be much worse than traditional force-directed algorithms. Tries to achieve the graph drawing principles in a higher dimensional space.
ACE	Like HDE, it is able to draw graphs quickly but a layout is not always able to be computed	Does well at computing graphs with few edge crossings		Nodes may end up being placed too close together	Especially with large graphs edge length uniformity is poor	Up to about 5000 nodes	Running times can grow as graphs get more challenging
ISOM	Known to be computationally efficient		✓	✓		Only tested on graphs up to 25 nodes	Map graph theoretic distances to Euclidean distances
Walshaw	Fairly fast, computed layout in less than 30 s for a graph with fewer than 10,000 nodes				✓	Given enough time can compute layouts for graphs with 100,000 nodes	Layout procedure is based on FR.
GRIP	Able to lay out graphs with more than 10,000 nodes in less than 1 minute		✓The projection from the higher dimensional space should aid symmetry.			Scales well to larger graphs	
FMS	Around 2 minutes for a graph with 6000 nodes but 10,000 nodes at the most	Not considered because of the use of KK's algorithm	✓Since based on KK and this was their main aim. Locally aesthetic.		Able to compute layouts with a uniform edge length	More than 6000 nodes	The graph should be aesthetic but specific criteria are not defined. Partly based on KK

(continued)

Table 1. (continued)

Algorithm	Performance	Graph drawing principles				Size	Notes
		Edge crossings	Symmetry	Even node distribution	Uniform edge		
FM^3	Not as fast as HDE but still quick and able to produce layouts for even very large graphs				Performs the most successfully out of FMS, ACE, HDE, FR and GRIP on edge length uniformity	Able to draw graphs with $>10{,}000$ nodes in <25 s and 100,00 nodes in <5 minutes.	Produces the most visually appealing layouts from FMS, ACE, HDE, FR and GRIP
Hu	Speed is similar to Walshaw and FM^3 and uses Barnes–Hut for optimisation	✓	✓	✓		Can lay out $>100{,}00$ nodes in <1 minute	Layouts are similar to Walshaw and FM^3 but can be more visually appealing than Walshaw's
OpenOrd	Has a parallel implementation to improve performance	✓	✓	✓	✓	Able to lay out graphs with $>500{,}000$ nodes	Adherence to the graph drawing principles is implied since the algorithm extends FR and simulated annealing but aims to produce a clustered graph
TopoLayout	Worst case complexity of $O(n^3)$	✓Crossing reduction for each topological feature				More than 70,000 nodes	Produces visually appealing layouts usually different to those produced by other algorithms

Table 2. A list of aesthetic considerations often used in graph layout and the motivation behind their use.

Aesthetic	Reason
Minimise edge crossings	Improves readability and aids the user in following paths. Edge crossings can also conceal important information and make a graph appear less approachable to the user[21]
Symmetry	Symmetry aids in the understanding of the structure of a graph
Uniform edge lengths	For a regular structure that prevents the graph becoming distorted
Uniform node distribution	For a regular structure, visual appeal and to prevent the graph feeling cluttered[22]
Separate non-adjacent nodes	Proximity implies a relationship and so adjacent nodes should appear more related than non-adjacent nodes
Node–edge overlap	Avoids visual elements appearing too clustered together and ambiguity as to where the edge ends[22]

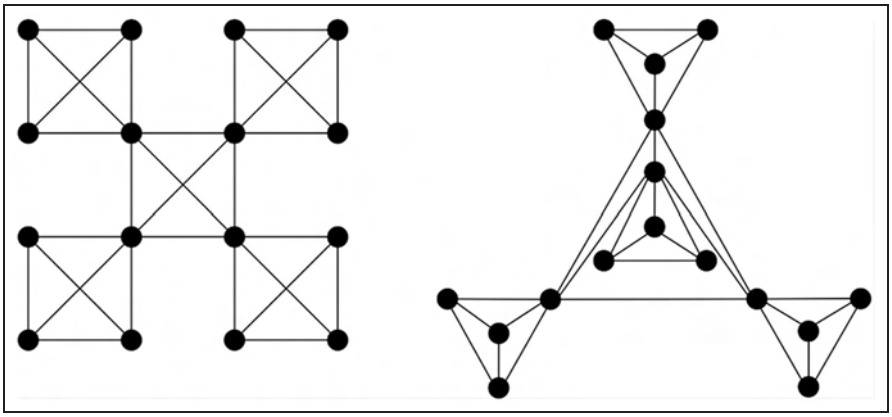

Figure 2. Two layouts of one graph. Both layouts are symmetrical, but users find the layout on the left easier to understand than that on the right, despite the fact that it breaks the principle of having no edge crossings. (Reproduced from Kamada and Kawai[23].).

about the view of the graph. Following this, it is not always the graphs that adhere to these principles which are the most comprehensible; in fact they can lead to graphs which are ambiguous or unintuitive to the user. The two graphs in Figure 2 provide an example in which symmetry is more important than the avoidance of edge crossings. Conversely, Purchase[24] conducted a user study on the effect of these principles on the understanding (with graph-theoretic type tasks) of the graph and found that minimising edge crossing was the most important criterion whereas symmetry was less so. This shows one of the problems with trying to use these principles for layout: no one is really sure which criteria are the most useful or in which situations they are most applicable.[17]

The designers of the graph drawing software tool Nicheworks[25] noted that trying to follow these graph drawing principles does not always create a better graph layout. Their software was designed to reflect edge weights, avoid the high computational cost of dealing with edge crossings and show clustering. However, most force-directed algorithms are popular because of their adherence to these principles and are generally evaluated on that basis.

There are two approaches to force-directed layouts: those based on Eades' spring-embedder and those which are solutions to optimisation problems. Eades' spring-embedded approach uses a spring-electrical system to find an equilibrium in the system so that the total force on each node is zero. The other approach, still inspired by Eades, treats the layout problem as an optimisation problem, which minimises an energy function (also known as the cost or objective function) designed with respect to the properties of the graph to be displayed.

Spring-electrical based approaches. Eades' spring-embedder[16] is the basis for almost all force-directed techniques. Nodes are modelled as steel rings and edges as springs; the system is put into a random initial configuration and released, leaving the system to reach

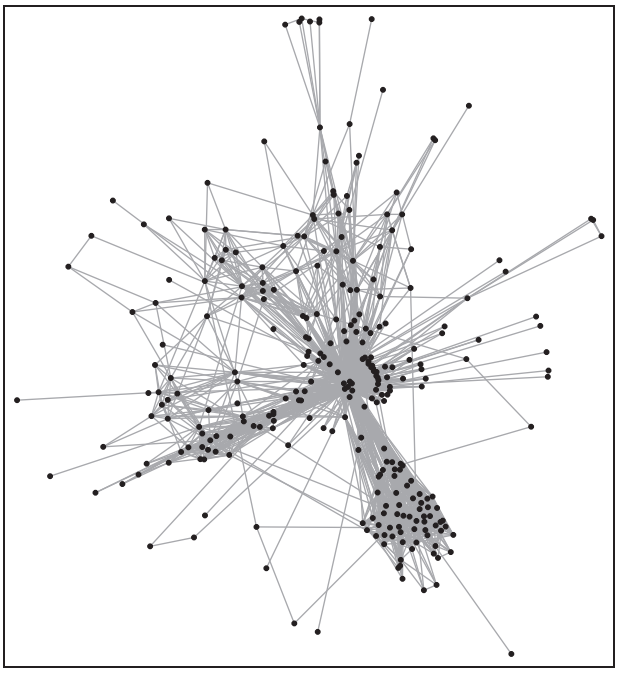

Figure 3. The spring embedder algorithm drawn using Cytoscape. The graph is a protein–protein interaction network with 283 nodes and 1749 edges and will be used as the standard graph for all running examples in this section unless otherwise specified.

a stable state where the force on each node is zero. The force on each node is the sum of attractive, $f_a = c_1 \log \frac{d}{c_2}$, and repulsive, $f_r = \frac{c_3}{d^2}$, forces on each node, where d is the length of the spring and c_1, c_2, c_3 are constants. From experimentation, Eades made the decision to use springs of logarithmic strength, claiming that linear strength springs were too strong. Connected nodes are attracted to one another whereas all other nodes, modelled as electrical charges, repel. The resulting layout should have edges of uniform length and symmetry. An example of spring-embedded layout is shown in Figure 3.

Despite its wide use, Eades[16] stressed his method's suitability only for graphs with fewer than 50 nodes with underlying structures such as grids, trees and sparse graphs rather than those with a dense structure. Therefore, it often produces poor layouts for large graphs and these problems are replicated in algorithms that build on Eades' approach.

The first adaptation, which was for speed and simplicity, of the spring-embedder algorithm was by Fruchterman and Reingold.[26] This was subject to the properties of connected nodes appearing close to one another (but not too close), aiming for a layout which conformed to the principles of even node distribution, few edge crossings, uniform edge length, symmetry and fitting the drawing to the frame. As with Eades, connected nodes attract while all nodes repel through a three-step process. The attractive, $f_a = \frac{d^2}{k}$, then repulsive, $f_r = -\frac{k^2}{d}$, forces are calculated, followed by the 'temperature' governing the distance each node can move during an iteration. d is the distance between the two nodes and $k = C\sqrt{\frac{\text{area}}{\text{numberofnodes}}}$ is the optimal distance between the two nodes where area is the space available and C is an experimentally determined constant.

The algorithm is not guaranteed to converge, so Fruchterman and Reingold suggest that 50 iterations of the algorithm is sufficient for an optimum layout. Their goal was for the algorithm to work well without requiring the user to tune various parameters to produce a satisfactory layout. They considered it to be fast, laying out most graphs in less than one second, but restricted these graphs to a maximum of 100 nodes. An example of a graph laid out using Fruchterman and Reingold's algorithm is shown in Figure 4(a).

Fruchterman and Reingold sped up their layout through a modification known as the Grid Variant Algorithm (GVA). The graph frame is divided into a grid and for each node the repulsive forces are calculated only between those nodes that are in the same or neighbouring grid squares; this was later altered to just being inside a specified radius. Repulsive forces are now

$$f_r = \frac{k^2}{d}u(2k-1) \qquad \text{where } u(x) = \begin{cases} 1 & x > 0 \\ 0 & \text{otherwise} \end{cases}$$

This speeds up the algorithm without affecting the quality of the graph and in some cases may improve the quality of the layout according to the graph drawing principles by preventing the nodes from becoming too widely spread, although Fruchterman and Reingold expected this modification to be useful only for larger graphs.

The algorithm is still used for graph layout with Genc and Dogrusoz[27] basing their model for laying out biological pathways in PATIKA[28] on it, and Garcia et al.[29] extending it to use as a basis for their layout for displaying gene ontology class structure of nodes in a protein–protein interaction network.

The final example of a spring-electrical-based system is the graph-embedder (GEM) algorithm. Frick et al.[30] wanted to produce a layout that conforms to the graph drawing principles for larger and more complex graphs but with 'interactive speed'. The exact principles are not explicitly defined but are said to be similar to the concepts for other spring layouts and the system is evaluated on edge crossings, edge lengths and node distribution.

GEM uses forces similar to those used by Fruchterman and Reingold, plus gravitational forces, which pull nodes towards the barycentre of their neighbours. Fruchterman and Reingold believed that a better cooling schedule could have significantly improved their algorithm and so Frick et al. implemented an adaptive schedule with local and global temperatures. The lower the local temperature of a node the closer it should be to its final position in the layout and the shorter the distance it can move in each iteration. A node's local temperature is dependent on its temperature in the previous iteration and whether the node is oscillating between positions or part of a rotating sub-graph. The global temperature is the mean of all local temperatures and the iterations stop when global temperature reaches a specified value or a fixed number of iterations have passed. The temperature is therefore adaptive to the state of the graph and, because of this, differs from cooling schedules such as those simulated annealing.[31] The quality of the layout should be similar to those of Fruchterman and Reingold and Kamada and Kawai ('Energy-based approaches'). This includes minimising edge crossings despite not being explicitly designed to do so. An example of the layout produced by GEM is shown in Figure 4(b).

Energy-based approaches. Energy-based approaches consider layout to be the minima of an optimisation problem in which an energy function encodes the desired properties of the graph. Solutions which result in local rather than global minima are not the optimal solution and, in this case, will not produce the optimal layout. As with the spring GEM algorithm, some energy-based layouts can temporarily move to higher energy configurations to reach the global minimum. The two main methods using this approach are from Kamada and Kawai[23] and Davidson and Harel's[31] simulated annealing algorithm.

Kamada and Kawai[23] use a spring approach with the key concept that Euclidean distance in the layout should approximate the graph-theoretic distance, i.e. the shortest path length between two nodes. The principle of symmetry is most important in this layout (see Figure 2) and the cost function represents the 'degree of imbalance' (lack of symmetry) in the layout and is based on Hooke's law (where the force exerted by a spring is linear and proportional to its displacement from its natural length). Optimisation requires solving partial differential equations based on the sum of the squares of the difference between the Euclidean and graph-theoretic distances of pairs of nodes. In each iteration only one node is moved and minimisation is carried out through Newton–Raphson iteration and the resulting layouts are symmetrical with few edge crossings. An example of a layout produced with Kamada and Kawai's algorithm is shown in Figure 5. There are similarities between this approach and those of multi-dimensional scaling, and these are further discussed in 'Dimension reduction for layout'.

Davidson and Harel's[31] simulated annealing also uses an energy-based approach. Annealing is the process of cooling a liquid slowly so that it forms a minimal energy crystalline structure. Their approach is for drawing graphs that comply with the graph-drawing

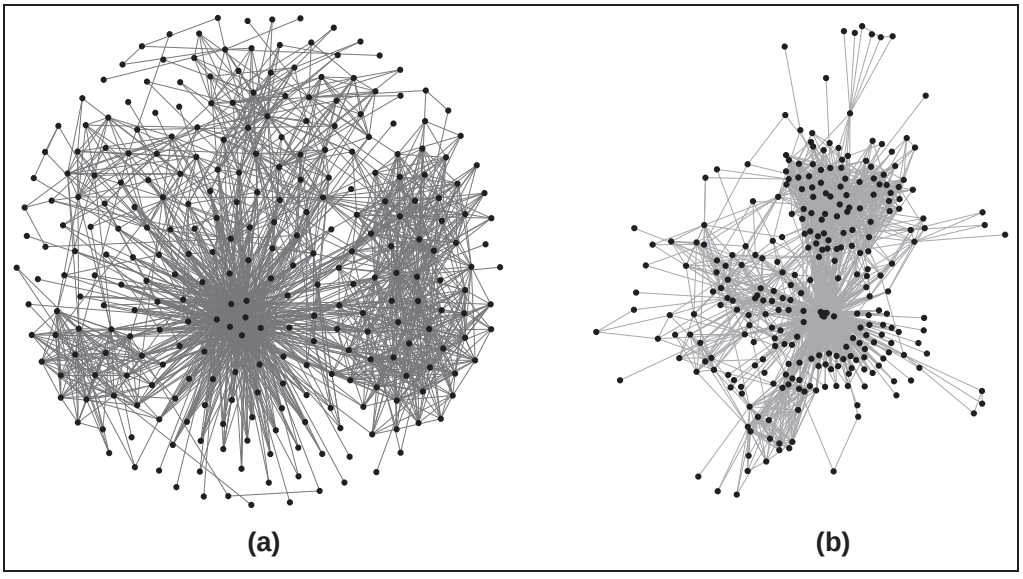

Figure 4. The protein interaction network laid out using Fruchterman and Reingold's algorithm (a) and the graph embedder (GEM) (b).

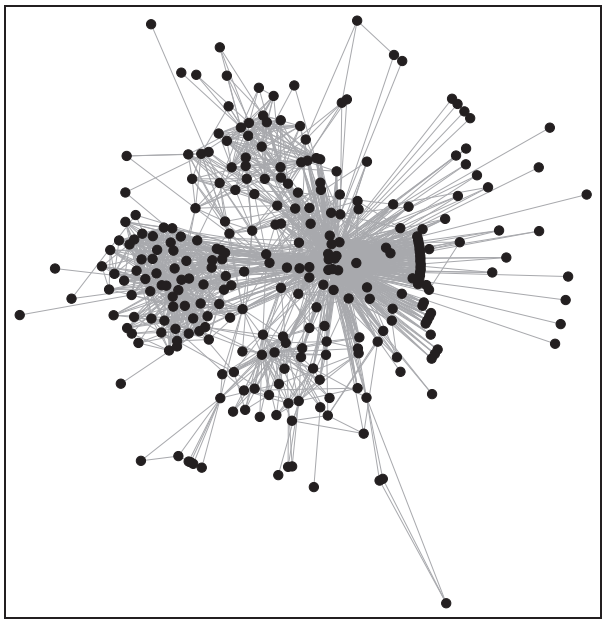

Figure 5. The protein interaction graph laid out using the Kamada–Kawai algorithm in Ondex.[32]

principles of evenly distributed nodes, uniform edge lengths and minimised edge crossings. They created an explicit cost function comprising parts which govern node distribution, borderlines (distance from the edge of the available layout area), edge lengths, edge crossings and near node-edge crossings, which can be weighted to prioritise specific aesthetics. The algorithm begins from an initial configuration and global temperature. In each iteration only one node is moved. The distance a node can move decreases with each iteration, and the temperature is recalculated at each step. The process continues until a termination condition, such as iterating through a fixed number of stages, is satisfied. Fine tuning of the graph can then follow.

Davidson and Harel admit that the algorithm does not perform well for graphs with over 60 nodes, but performance is similar to the models by Fruchterman and Reingold, Kamada and Kawai, and GEM. They also concede that the algorithm was coded for functionality rather than performance; consequently, the algorithm is often not used practically because it is too slow to be deemed useful.[19] Exceptions to this are from Li and Kurata,[33] who implemented it in their CADLIVE system for laying out biochemical networks, the jGraph[34] layouts of Cytoscape[35] and its use in OpenOrd.[36]

Three other notable energy-based techniques are from Tunkelang,[19] Noack's LinLog layouts[37,38] and ForceAtlas[39] from Gephi.[40] Tunkelang's initial approach is distinctive in that it requires computation of a minimum spanning tree (a tree which connects all nodes of the graph) to decide the order in which nodes are placed. Optimisation of node position is based on what he calls the aesthetic cost function encoding the principles of uniform edge lengths, even distribution of nodes and minimal number of edge crossings. It follows the approach of Fruchterman and Reingold in using a grid to speed up computation of repulsive forces. It also outperformed simulated annealing and Fruchterman and Reingold in terms of minimising edge crossings for all graphs, while for edge lengths and node distribution Tunkelang's approach performs better on sparse graphs with Fruchterman and Reingold's better for denser graphs.

Noack[37,38] takes an alternative to the principles-based approach as his main aim is to highlight clustering in a graph. Other force-directed algorithms usually miss this by striving for uniform edge lengths, but longer edges are required to separate clusters. The weak connection between using graph-theoretic distance and identifying low coupling between parts of the graph also hinders clustering.[38]

Noack proposes two models, both with linear attractive forces between adjacent nodes and logarithmic repulsive forces. In the node repulsion model repulsive forces are calculated between all nodes in the graph, again using their Euclidean distances, whereas in the edge repulsion model repulsive forces act on edges by weighting the calculation according to the degree (number of edges connected to the node) of each node. The edge model removes the node model's bias towards attraction by ensuring that nodes which are strongly attracting are also strongly repelling; similarly for those nodes with weak attraction. Therefore, nodes with a high degree are less likely to be clumped in the centre of the graph and, in comparison with Fruchterman and Reingold's layout, it is much more clearly able to show any underlying clustered structure in the graph.

The final energy layout has strong associations with Noack's. ForceAtlas[39] was developed for use in Gephi[40] as their users were dissatisfied with given layout algorithms. Rather than a layout, Jacomy et al. call their method a 'spatialization', which is defined as the act of projecting data onto space. The method aims to optimise the speed versus precision approximation. The forces in the algorithm are between Noack's edge repulsion model (logarithmic attractive force, linear repulsive force) and Fruchterman and Reingold's layout (quadratic attractive, linear repulsive force) with both linear attractive and repulsive forces. Attractive forces are again distances between nodes and repulsive forces are based on distances and node degree plus one. This ensures that all nodes have at least some repulsive force, unlike Noack's model, and so poorly

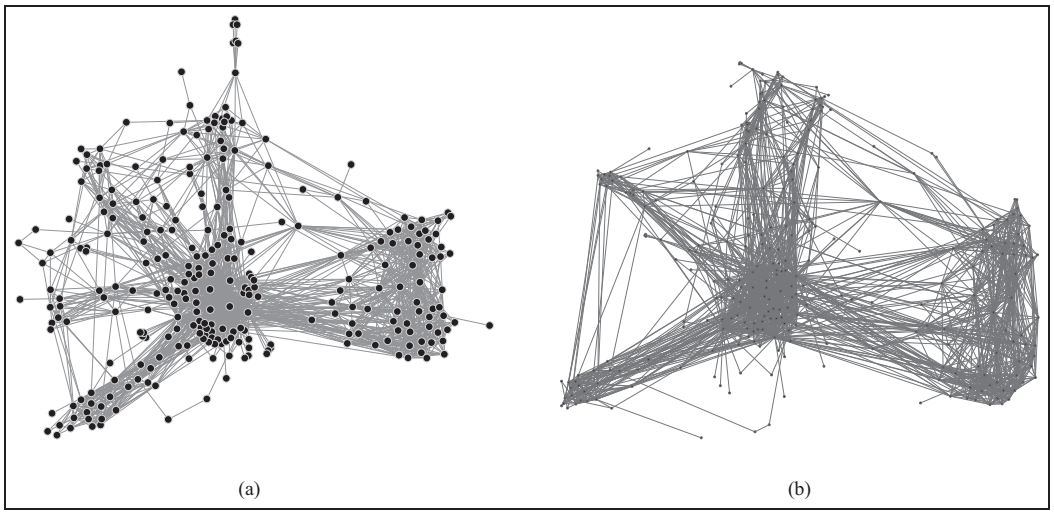

Figure 6. The protein interaction network laid out using the ForceAtlas layout. The layout on the left is the ForceAtlas style only and on the right the layout uses Noack's LinLog algorithm to aid the clustering of nodes. Both algorithms produce layouts with a similar shape, with the LinLog adjustment resulting in a layout that requires a greater area.

connected nodes are brought closer to well-connected ones reducing visual clutter. The algorithm maximises speed until it is clear that some nodes are unstable, at which point it slows and emphasises precision. Jacomy et al. also particularly wanted the layout to be interactive for the user and so there are many options that can be configured even while the layout is running, which may alter the layout but also give users a better understanding of how the algorithm can be manipulated by the user to produce a layout that is most suitable for them.

In general, ForceAtlas produces better quality (in terms of a normalised edge length metric proposed by Noack[41] based on size and graph density) with fewer iterations for most graphs than Fruchterman and Reingold and Yifan Hu (see 'Computational improvements') and the results were even more convincing when the technique was combined with Noack's edge repulsion model; a comparison of the ForceAtlas layout with and without the LinLog adjustment can be seen in Figure 6. Performance improvements have also been made by utilising the Barnes–Hut algorithm[42] (explained further in 'Computational Improvements') and multithreading.

Dimension reduction for layout

Dimension reduction is the process of taking data expressed in high-dimensional space and projecting it onto a lower-dimensional space. The challenge is to try to retain the information that is in the high-dimensional space and capture it in the lower-dimensional representation. Most dimension reduction techniques currently used for graph layout use the graph-theoretic distance between a pair of nodes as the information that is to be preserved. This section provides an overview of dimension reduction techniques in graph drawing, in particular multidimensional scaling (MDS), linear dimension reduction and self-organising graphs.

MDS involves minimising the difference between the Euclidean and graph-theoretic distances. There are two approaches of MDS to solve this problem: distance scaling and classical scaling.

Distance scaling is more common and the idea is to directly compute an approximation of the difference (the dissimilarity) between the graph-theoretic and the Euclidean distance for each pair of nodes in the layout. The sum of the squares of the difference is called the 'stress' of the layout and the aim is to minimise this stress through an optimisation procedure. If the stress is considered to be too high then this can be taken as an indication that the layout is not an accurate representation of the original dissimilarities.[43] This stress function is regarded as being almost identical to Kamada and Kawai's energy function for force-directed layouts. The difference between the two is that Kamada and Kawai use Newton–Raphson iteration to find the minima whereas distance scaling minimises the stress through the statistical technique of stress majorisation.[44]

Distance scaling was first used for graph drawing by Kruskal and Seery[45] for social network layout and since then it has also been used by Freeman[43] to show relationships between workers in a department store. In addition, Buja et al.[46] have implemented it as part of the XGvis system.

With distance scaling the optimisation procedure may result in a local minima solution. However,

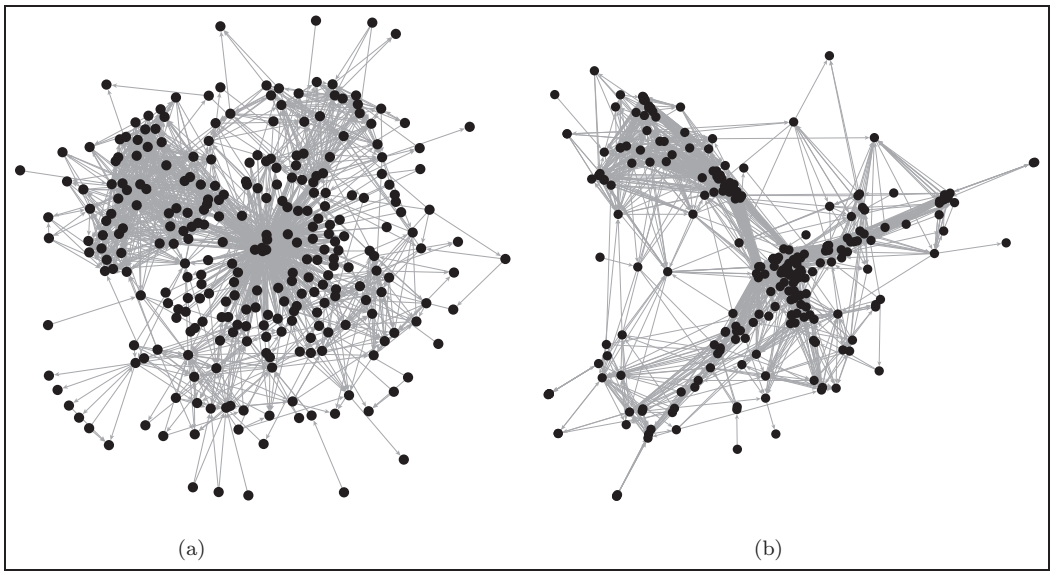

Figure 7. The distance (a) and classical scaling (b) layouts for the protein interaction network.

classical (or Torgerson–Gower) scaling follows a method of fitting inner products to the dissimilarities and finding an exact solution through the spectral decomposition. Formally, the spectral decomposition is the representation of the matrix of graph-theoretic dissimilarities between node-pairs as the matrix of its eigenvectors and a diagonal matrix of corresponding eigenvalues. By choosing the largest d eigenvalues (where d corresponds to the number of dimensions one wants to reduce to) the lower dimensional representation of Euclidean distances between node-pairs can be found.

Brandes and Pich[47,48] proposed that classical scaling should provide a good alternative to distance scaling, but its quadratic running time is prohibitive to implementation. They proposed a sampling approximation technique known as Pivot MDS. In Pivot MDS some nodes are assigned to be pivots (the first pivot is chosen randomly and subsequent ones are chosen by being the node with the greatest graph-theoretic distance from all selected pivots so far). Non-pivots are positioned based on their distances to the pivot nodes whereas the pivot nodes can use the distances both to other pivot nodes and non-pivot nodes to determine their position. This is able to approximate the results of classical scaling but in linear time. Brandes and Pich also recognised that selecting the number of pivots was a key problem and so implemented a second method, progressive MDS, in which the number of pivots required could be defined along the way.

Brandes and Pich[48] recognise the importance of distance scaling, noting that it is more successful at showing local details than classical scaling, which is more suited to capturing global structure. They recommend using a version of classical scaling for initial layout followed by refinement through weighted distance scaling (in fact they recommend this refinement for any layout method). They also warn about the merits of approximating Euclidean to graph-theoretic distances, pointing out that if the structure of the graph is not well captured by these distances then whichever layout method is used the resulting layout will not be good. Example layouts produced through both distance and classic scaling are shown in Figure 7.

Although the technique uses principal component analysis (PCA) for dimension reduction, Harel and Koren's[49] high dimensional embedding (HDE) has strong similarities with Pivot MDS. The aim of this layout was to follow the convention of placing adjacent nodes close together and non-adjacent nodes further apart. The graph is first embedded in a high-dimensional space by choosing 50 nodes as pivots and associating each pivot with a dimension; nodes are then expressed in high-dimensional space as graph-theoretic distances from pivots. Then the graph is linearly projected onto two dimensions using PCA (although projection onto any lower dimensional space is possible). PCA maps the data onto the first two principal components, which account for the most variance in the data. HDE is one of the quickest layout techniques available and is implemented in both Topolayout[50] and Visone,[51] although it does seem to come at the cost of quality, in terms of edge lengths, co-located nodes and edge crossings.[52] Brandes and Pich[48] note that both algorithms have similar running times, although the quality of Pivot MDS is higher than HDE. Koren[53] improves upon the HDE method by replacing PCA

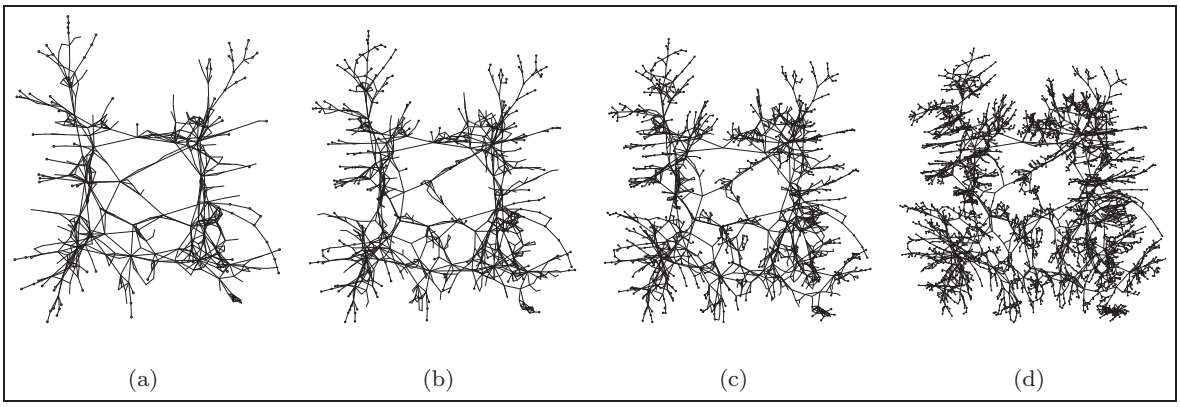

Figure 8. Four stages of a multi-level algorithm produced using Hu[61] in Gephi.[40] Each image, from left to right, shows the layout propagated back up through each less coarse version of the graph, i.e. (a) is the coarsest version of the graph while (d) is the final layout.

with subspace optimisation, the aim of which is to find a subspace of low dimensionality that can display the graph with what they call a 'nice' (short edge lengths and uniform node distribution) layout.

Spectral graph drawing also reduces dimensions by using the eigenvectors associated with the two largest eigenvalues of the graph-theoretic distance matrix found via the spectral decomposition.[54] The method is also sped up by using only a sample of the nodes to compute the graph theoretic distances, much like HDE and Pivot MDS.[55]

Self-organising maps (SOM) can be used to draw a self-organising graph. Here an unsupervised neural network is formed to project high-dimensional data onto a lower-dimensional space. For each node its neighbourhood is defined to be all connected nodes. There exists a set of two-dimensional, uniformly distributed training vectors and so each node is also described as a position in 2D space. The system tries to learn the distribution of the training vectors by selecting a training vector followed by its closest node. This node and all its neighbours' positions are updated by moving the nodes closer to the position of the training vector. This method tends to be easy to implement and efficient because it has no computationally expensive iterations. Bonabeau's[56,57] method can be used as a standalone graph layout or as a pre-processing step to provide initial layout. Owing to the difficulty in training smaller graphs the method actually works better for larger graphs. Meyer's[58] inverted self-organising map (ISOM) fits graph-theoretic to Euclidean distances to lay out the graph using an SOM method utilising their strong clustering and structure detection capabilities.

Computational improvements

While force-directed techniques are generally not suitable for graphs with node numbers in the hundreds or thousands, when evaluated qualitatively or against the aesthetic principles on graphs with up to 40 nodes, they do give good results.[19,30] Above that size, multi-level (or multi-scale) algorithms are one option that can be used to make force-directed techniques more efficient.[37] The idea behind multi-level techniques is to find a sequence of coarser representations of a graph, optimise the drawing in the coarsest representation, and propagate that layout back through to the original graph. The coarser representations are created by collapsing connected nodes whose edges becomes the union of the edges of all the nodes it comprises.

The inspiration for using a multi-level technique for graph drawing came from Hendrickson and Leland,[59] although the original idea came from particle physics.[60] Their idea was to use a multi-level technique to solve the graph partitioning problem of how to divide the nodes of the graph into sets such that there are as few edges crossing between the sets as possible. Hendrickson and Leland solved this problem using the coarsest representation of the graph to define the partitions. An example of the stages of a multi-level graph layout algorithm can be seen in Figure 8.

Once the graph has been reduced to its coarsest level the method to actually lay out the coarse graph varies. This can be using a force-directed algorithm,[60,62–64] a dimension reduction approach[65] or a spectral approach.[66,67] Computational improvements are not limited to multi-level methods and other techniques are also successful.

Multi-level methods vary not only by the layout algorithm used but also by the coarsening scheme implemented. Cohen[65] developed the first multi-level technique and it has similarities to GRIP.[60] The algorithm follows the recurring theme that graph distances should be reflected in Euclidean distances by using an incremental layout approach with MDS. Instead of

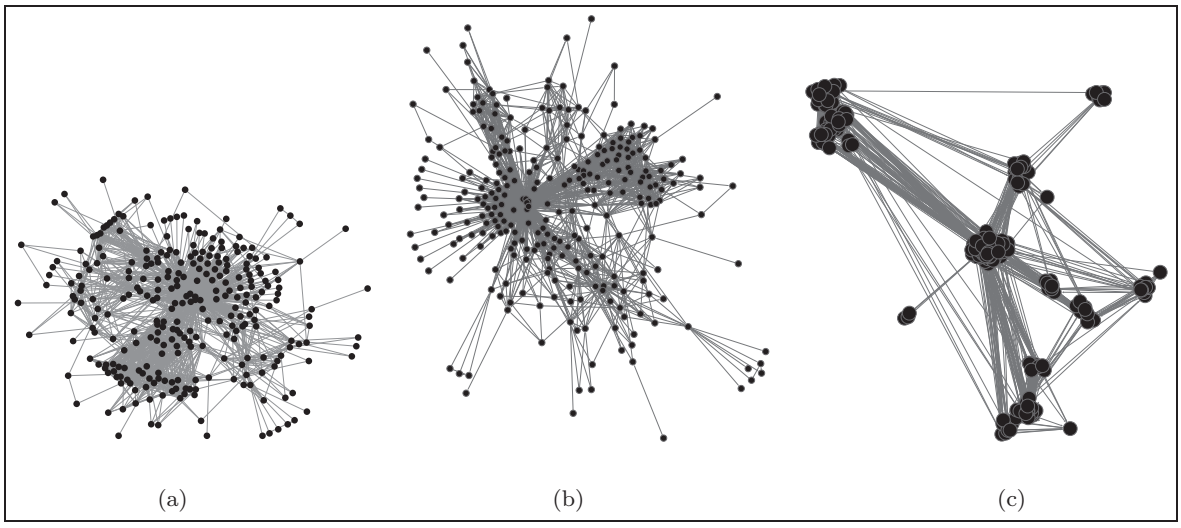

Figure 9. Three layouts of the protein interaction network produced using algorithms that aim to provide computational improvements over traditional force-directed methods (a) is FMS, (b) is Yifan Hu's and (c) is OpenOrd.

modelling the Euclidean distances on graph-theoretic distances between nodes, Cohen opts to use the linear-network distance. The linear-network distance reduces the distance between nodes if there are multiple paths between them, further emphasising the structure of the graph; in particular, this strategy makes clusters in the graph more prominent in the drawing.

Walshaw,[64] GRIP (Graph Drawing by Intelligent Placement)[60,68] and Hu[61] all share the idea of using maximum independent vertex sets (MIVS) for their coarsening procedures where an IVS is formed if there are no two nodes connected by an edge in the subset, and it is said to be maximal if the addition of any edge to the subset would break this property. One of Hu's[61] contributions is a hybrid approach that allows the algorithm to follow a simple edge collapsing coarsening scheme typically but if more than 50% of the nodes remain after the edge collapsing then the algorithm reverts to the MIVS coarsening scheme. FMS's (fast multi-scale method)[63] coarsening method is based on an approximation to the k-centres problem-forming clusters, which are then shrunk to single nodes while its predecessor[69] creates the hierarchy based on cluster number, degree number and homotopic number. In FM^3(fast-multi-pole, multi-level method) Hachul and Jünger[62] use a novel method based on solar systems, in which each node is classified as a sun, planet or moon and each solar system is collapsed to one node.

Many of the multi-level algorithms are variations of the force-directed layouts and since force-directed layouts are modelled as physical simulations they can be described as n-body problems. So, along with using a multi-level scheme to improve computation, a Barnes–Hut[42] simulation can be used with a quad-tree to improve the order of complexity of the calculation from $O(n^2)$ to $O(n\log n)$. A square is placed over the initial layout of the graph and is divided into four. If there is more than one node inside a sub-square then it is divided into four. This procedure continues recursively until each square contains at most one node. Nodes are then clustered based on their position in the quad-tree and those clusters considered to be far from the node of interest form a super-node whose forces can be considered as one reducing the complexity of the force calculation.

In terms of the force-directed methods used in multi-level algorithms, Walshaw's method is quick and is able to lay out a graph of 500,000 nodes in only a few minutes using a modified version of Fruchterman and Reingold's algorithm. GRIP initially uses Kamada and Kawai's algorithm followed by Fruchterman and Reingold's for refinement, but the difference with GRIP is that the layout can be done in any dimension and Gajer et al. then suggest projecting down to two dimensions at the end to create a smoother layout. FMS (Figure 9(a)) also uses Kamada and Kawai's layout technique aiming to conform to the graph layout principles at both the local and global levels. Hu[61] (Figure 9(b)) claims to produce layout with a similar speed to Walshaw but with better results for certain graphs. He utilises the Barnes–Hut approximation but can limit the number of recursive divisions, and adds an adaptive cooling scheme to a general force-directed model where step length remains constant until there are five consecutive energy reductions, in which case the step size is increased, or if there is an energy increase then step length is decreased. FM^3 is the final multi-level force model which also uses the quad-tree

to approximate repulsive forces by rapidly evaluating potential fields on each node. Further computational improvements of FM^3 have involved implementing a version of it on the GPU for speed resulting in layouts that were at least 20 times and up to 60 times faster.[70]

Spectral graph drawing approaches can also use eigenvectors of the Laplacian matrix to produce a projection of the layout via the spectral decomposition. For a general undirected graph the Laplacian matrix is symmetrical with node degrees along the diagonal, zero where there is no edge between two nodes or the negative weight of the edge between the two nodes if there is. If the graph is not weighted all edges can be taken to have equal weight.

The algebraic multi-grid computation of eigenvectors (ACE)[66,71] algorithm is able to draw graphs with millions of nodes in under one minute. It does this by stating Hall's[72] placement algorithm as the eigen-projection problem as in Equation (1), where L is the Laplacian matrix, x is a vector of coordinates of the position of each node and $1_n = (11\ldots1)^T \in R^n$. ACE then requires the coarsening phase, known as the algebraic multi-grid technique, to simplify and solve the problem. Here the coarsening phase is based on stating the problem initially in a high dimension and then to keep restating the problem in lower dimensions. Once expressed in the lower dimension the problem can be solved in that dimension and the solution can be projected back up through the increasingly less coarsened graphs to the original problem.

$$\begin{aligned} &\min_x x^T L x \\ &\text{given } x^T x = 1 \\ &\text{in the subspace } x^T \cdot 1_n = 0 \end{aligned} \tag{1}$$

Frishman and Tal[67] also suggested using a spectral approach for layout, which used a sequence of coarsened graphs that were then partitioned. Layouts for each partition, followed by a layout for the whole graph, were found. This showed a greater clustering ability than GRIP and sped up their technique further by also implementing it on the GPU.

OpenOrd[36] is a graph layout based on the VxOrd[73] algorithm and implemented in the graph drawing software Gephi. It is specifically designed to uncover global structure, improve the visual appeal and shorten the running time of force-directed graphs. It is based on the phases of simulated annealing and incorporates node clustering by ignoring or cutting long edges in the layout (because of the force calculation, domination of the repulsive force gives more clustering). The multi-level part uses Walshaw's algorithm but instead of using random initial positions for coarsening, average-link clustering is used, which makes use of both edge weights and graph distance to find clusters to collapse. The algorithm can be implemented in both serial and parallel. They have tested the layout on real-world datasets with over 500,000 nodes and found that the serial and parallel implementations produce similar results that were more visually appealing than VxOrd. As can be seen from Figure 9(c) the OpenOrd algorithm produces a layout markedly different to the others, clearly showing an overview and clustered structure of the graph.

Chen and Buja[74] also use a combined approach in a method which can be applied to dimension reduction as well as to graph drawing problems. They do this by taking a force-directed function and modelling the attractive and repulsive forces with box-cox transformations, which results in a generalisation of many previously published energy functions in graph drawing such as Noack's LinLog. In this case the parameter (such as clustering) that we wish to optimise in the algorithm can be tuned.

Topolayout[50] is a completely different approach to finding coarser representations of the graph whereby the algorithm detects topological features in the graph (such as trees, clusters and complete graphs) and collapses them into a single node. This is done repeatedly until the coarsest graph is reached and then each feature is assigned its own layout algorithm and laid out accordingly. These include circular layouts, the GEM algorithm and also detecting and using the high-dimensional embedding (HDE) layout algorithm discussed in 'Dimension reduction for layout'. In their evaluation they found it performed better, in general, than FM^3, ACE and HDE in terms of speed and a visual assessment of the quality of the layout produced.

Evaluation of algorithmic layouts for graph visualisation

Force-directed, dimension reduction and multi-level algorithms are often evaluated to determine which gives the 'best' layout. Typically, it is the adherence to some of the principles of graph drawing, along with the time taken to draw the graph, which are used to determine which layout is best, although sometimes only a visual inspection of the graph is used to assess visual quality, as in Hachul and Jünger[75] and Archambault et al.[50]

The criticism of force-directed methods is wide and varied but that does not seem to have prevented them from becoming popular. The two main criticisms are that they have a high running time, and therefore become unsuitable for large graphs, and the second is their tendency to get trapped in local minima (even with some improvements added).[60] These problems

are magnified in larger graphs resulting in layouts with many edges crossing, node occlusion and labels on nodes which are unreadable. Ultimately this leads to nodes being placed in arbitrary positions[76] or layouts which have led to the term 'giant hairball' being applied, an example of which can be seen in Figure 10. They are also often unpredictable and may produce very different graph layouts for graphs which differ only slightly in structure,[10] a problem known as 'preserving the mental map'.[77]

However, not all algorithms perform equally, so a number of comparisons between layouts measuring running time and other criteria have been conducted. Tunkelang[19] was the first to perform a quantitative evaluation of his layout, comparing it with Fruchterman and Reingold's and simulated annealing. He evaluated on the principles of uniformity of edge lengths, uniformity of node distribution and number edge crossings on a range of graphs. Small graphs had about 16 nodes and large graphs between 50 and 60. Sparse graphs have nodes with degree at most three and dense graphs with degree greater than four. In both small and large sparse graphs Tunkelang's method outperformed Fruchterman and Reingold's and simulated annealing on all three measures. On the dense graphs Fruchterman and Reingold's produced a graph with more uniform edge lengths and node distribution than Tunkelang's. Simulated annealing gives the worst performance out of the three.

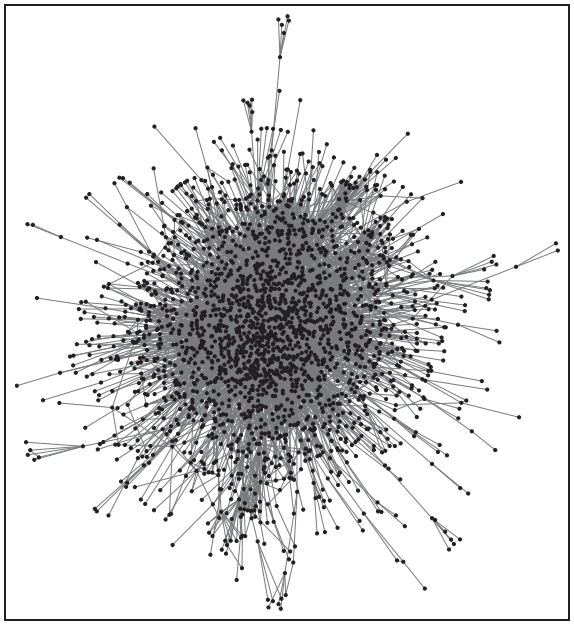

Figure 10. A larger protein interaction network (approximately 1800 nodes) laid out using the GEM algorithm. The algorithm produces a layout for the graph, which would be termed a giant hairball.

Frick et al.[30] evaluated their GEM algorithm against Fruchterman and Reingold's and Kamada and Kawai's on 30 graphs on the criteria of running time and the quality metrics of edge crossings, mean edge length, edge length deviation and node distribution. GEM was always the fastest algorithm, being at least four times faster that Fruchterman and Reingold's. Kamada and Kawai's layout slowed considerably once graph size was larger than 30 nodes. In terms of the principles the three algorithms produced similar results, though GEM fared slightly better than Fruchterman and Reingold's while Kamada and Kawai's did better than GEM on the aesthetic criteria with the smaller, sparse graphs.

Brandenburg et al.[78] compared Fruchterman and Reingold, Kamada and Kawai, GEM, simulated annealing and Tunkelang's approaches on general, undirected graphs with straight lines. They found that most graphs drawn conformed to the aesthetic principles of graph drawing, in particular uniform node distribution and edge length, while also stating them to be empirically visually appealing. The layouts were also stable, producing much the same graph each time the chosen algorithm was run. However, the graphs were small with no graph having more than 150 nodes plus edges in total. Fruchterman and Reingold, Kamada and Kawai, GEM and simulated annealing all produced relatively similar graphs, especially showing symmetry, whereas Tunkelang's generally produced a different drawing with no symmetry. They suggest using Tunkelang's method as an option if the other methods are unable to produce an adequate layout. Kamada and Kawai's and GEM were found to be the quickest, followed by Fruchterman and Reingold's if the graphs were kept small. Simulated annealing is a flexible method because of how it can be weighted to prioritise specific aesthetic principles, but a lot of time and patience is needed to configure these parameters and then run the algorithm. Ultimately, they recommend that Kamada and Kawai's or GEM should be used first, followed by Tunkelang's and then simulated annealing.

Hachul and Jünger[52,75] evaluated the same six layout algorithms twice; they were Fruchterman and Reingold's grid variant algorithm (GVA), FM^3, GRIP, FMS, HDE and ACE. Firstly, they evaluated only on the basis of run time and their own visual inspection. A total of 28 graphs were chosen on which to perform the evaluation and were classified as artificial or real-world and kind or challenging. In general, GVA was always the slowest algorithm and took over 5 hours to compute for the largest graph. FM^3 was generally faster and GRIP up to nine times faster again. FMS produced layouts within a similar time period to FM^3, but it is restricted to graphs of fewer than 10,000

nodes. Except on the two most challenging graphs, ACE computed layouts in less than 10 seconds and HDE computed most in under 1 second and all in under 5 seconds. In inspected visual quality terms they considered GVA to always produce a poor layout with all other methods able to produce visually pleasing results for the kind graphs. Despite being the quickest, ACE and HDE suffer from node occlusion. GVA, FM^3 and HDE were the only algorithms able to produce layouts for all graphs and they considered that FM^3 always produces a pleasing layout.

In Hachul and Jünger's[52] second evaluation they used the same layouts and test bed of graphs but replaced the visual assessment with quantitative measures of uniform edge length, number of edge crossings and non-overlapping of nodes and edges. For the kind graphs, GVA, FM^3, FMS, HDE and ACE all produced layouts with uniform edge length; however, ACE and HDE produced much more variation for the challenging graphs. In terms of edge crossings all graphs performed better than GVA and particularly ACE and then FM^3. FMS and HDE produced graphs with a range of edge crossings while those plus FMS also gave layouts with many overlapping nodes and edges. Again they reported FM^3 to be the most generally pleasing layout in terms of complying with the graph drawing principles. Ultimately, the authors recommended that when attempting to lay out a graph HDE should be used first, followed by ACE, but if neither produces acceptable results then FM^3 should be used.

As mentioned at the end of the 'Energy-based approaches' section, Jacomy et al.[39] compared the ForceAtlas layout with the ForceAtlas layout with a LinLog implementation, Fruchterman and Reingold's layout and the layout produced by the non-multi-level implementation of Hu.[61] ForceAtlas in general produced better quality graphs (in terms of a normalised edge length metric[41]) in fewer iterations. When combined with the LinLog implementation, ForceAtlas, for long-term quality (letting the algorithm run for 750 iterations), was shown to produce the highest quality in terms of Noack's metric. This is perhaps not a surprise considering the LinLog layout was developed for this optimising metric.

There are few comparisons of the dimension reduction techniques against other algorithms but a comparison from Brandes and Pich[48] between HDE, FM^3, GRIP and classic MDS notes that each of the layouts can be improved by applying up to 60 iterations of distance scaling to the layout as a post-processing refinement to any original algorithm.

User studies of the effect of layout are less common, but one exception to this is the comparison of three force-directed layouts by Purchase.[24] She found that for three layouts tested (Fruchterman and Reingold, Kamada and Kawai, and Tunkelang) there was little variation in users' understanding of the graph between layouts despite the fact the each layout emphasised a different aesthetic principle. This could indicate that the use of these criteria in layout is not as important as first thought; however, the graphs tested had only 17 nodes and 29 edges and so the graphs may not have been large enough distinguish between the principles, the principles were competing with each other or the graphs were too small to produce much variability in the layout or difficulties to overcome.

What is interesting, though, is the type of tasks she asked her users to complete. These were finding the shortest paths, identifying nodes to remove in order to disconnect the graph and identifying edges to remove in order to disconnect the graph. A particular consideration is whether these tasks are representative of those carried out by users in visual network analysis.

The readability and graph drawing principles qualities emphasised by most force-directed methods seem to support this style of task, and are feasible on the small graphs for which force-directed layouts work optimally, but it is unclear whether these types of accurate, precise measurements are typical analysis tasks for graphs with hundreds or thousands of nodes. Therefore, if force-directed layouts are optimal for a particular task and these tasks are completed only on a certain size of graph then extending force-directed techniques to larger graphs would produce a layout that is only perceived to be optimal for those tasks. So if those kinds of tasks become infeasible because of the volume of nodes and edges then the better layouts should support the user for a different set of tasks.

Noack's[38] LinLog layout is optimised for clustering to give a clear representation of the structure in the network, and van Ham and Rogowitz[79] found users tried to optimise clustering ahead of any other aesthetic metric, which also indicated that users are more concerned with overall structure. Another aim for layout should, then, be to support users in tasks concerned with overview, structure, exploration, patterns and outliers.[80] Algorithmic layouts which tend towards this motivation for layout are Noack's LinLog and OpenOrd.

Away from direct comparisons, a general problem noticed by Gansner and North[81] is that most force-directed algorithms were drawn with 'point' nodes in mind (nodes that are drawn as a dot rather than a disc) whereas many graphs in information visualisation require nodes to be shown with varying size, colour and shape, and for them to be labelled or interactive. Archambault et al.[50] call this being 'area-aware'. This is particularly prevalent in network visualisations, where being able to interact with and explore the graph

is more common than with graph drawing only. An alternative to simply scaling up the diagram is to use a post-processing solution of constructing a Voronoi diagram over the graph. A Voronoi diagram is constructed by partitioning a plane into convex polygons such that there is one node in each polygon known as a cell. Nodes are then moved to the centroid of their cell, which is the point most removed from all other nodes; thus, they no longer overlap but neither do they lose the relative distance between each other. Additionally, the straight line edges are then replaced with smooth curves to prevent node-edge overlaps, which has been found to improve readability.[82,83]

Running time aside, it is still the graph drawing principles on which most graph layouts are judged. However, studies on user requirements and the impact of these principles are lacking, especially on graphs of any significant size. This means that the layout algorithms used may not be satisfying anyone's requirements at all.

Because of this dissatisfaction with standard layout outputs, new techniques have been developed. These techniques often make use of node-attribute data. Node-attributes are additional pieces of data that may be known about each node, which can be used to enrich the understanding of the graph. The evaluation of these techniques is more focused on what can be learnt or what insights can be made through using this layout than on conforming to specific principles. Techniques that take advantage of node-attributes are discussed in the next section.

Using node-attributes for layout

Graphs that are defined along with attributes from their nodes are termed multi-variate graphs. Graphs that have this kind of structure are actually quite common.[35,43,84,85] The most popular way of visualising these underlying data is to use retinal variables such as the colour (and colour gradients), shape and size of the drawings of the nodes; additionally, glyphs can also be used.

Here we take an attribute to mean:

(a) a piece of data about a node (or edge) that already exists;
(b) a derived item of data about a node such as a computed centrality metric or a cluster generated from an algorithm; or
(c) a user-defined restriction that they want the graph to portray.

As stated above, users strongly interpret the proximity of a node in relation to others as the existence of a relationship or similarity between two nodes. Similarity of attributes would also imply a similarity between two nodes. Therefore, introducing attributes is one way of relating the layout of the graph to the properties of its nodes. Being able to make inferences about correlations between the structure of a graph and node-attributes is one way of increasing the potential insight to be gained from visualising it.[86]

Using attributes to influence the layout of the graph is not such a recent idea. As far back as the 1930s Jacob Moreno realised that 'variations in the locations of points could be used to stress important structural patterns in the data';[87] for example, simply separating boys and girls when graphing friendships between classmates or placing American football players in their positions on the field in a pass network gives valuable insight into the structure of relationships in the graph.[88] Lundberg and Steele[89] and Northway[90] used sociometric status for determining node position. Lundberg and Steele used those with high status as the nuclei of the network and placed others in a ring around them for presenting their research on social patterns in a village. Northway used a layout of concentric circles, known as a target sociogram, putting nodes with higher values close to the centre for displaying relationships between pupils in a classroom. A similar method has also been used by Brandes and Wagner[51] and Brandes et al.[91] to show the preventative measures for HIV introduced by local organisations in Germany where various node centrality measures were used to show status.

Recently, it has become even more common to use node-attribute data for layout of graphs. There are three main ways in which attributes can be used in graph layout: one is to impose a set of restrictions on the placement of nodes; a second is to use membership of a group or cluster to position the nodes; and a third is to directly map an attribute (or attributes) to a coordinate in the layout space (e.g. x and y in a Cartesian system).

Constraint-based layouts

As with multi-level techniques, constraint-based methods often make use of force-directed algorithms.[7] As shown in the section 'Family of force-directed and related graph drawing algorithms', force-directed algorithms and their derivatives use aesthetic principles as motivation, in their implementation and for their evaluation. Constraint-based techniques impose some user-defined placement criteria on all or a selection of the nodes in addition to using a layout algorithm. These can be constraints such as fixing node position, separating certain groups of nodes or adding layout

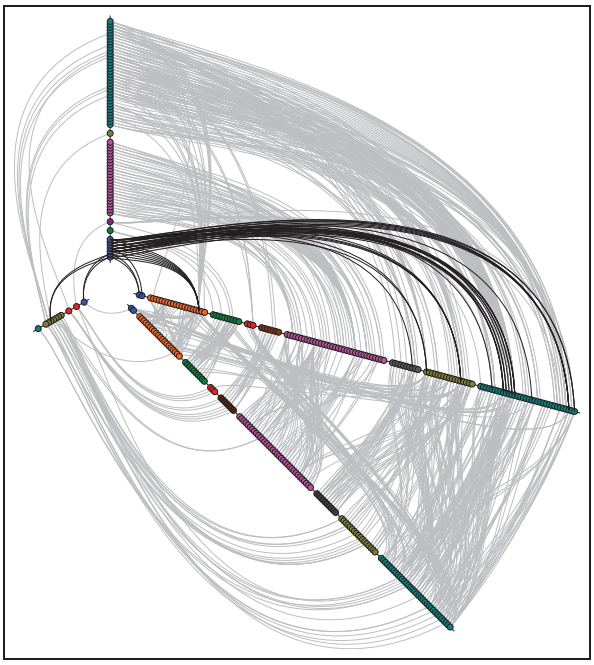

Figure 11. A hive plot showing dependencies between classes of the Flare visualisation tool-kit implemented in d3.js. (Image based on http://bost.ocks.org/mike/hive/).

constraints to a fixed sub-graph. These constraints often come from node-attributes in the data.

Probably the most well-known use of constraints in graph drawing is the Sugiyama et al.[92] approach for hierarchical structures. The hierarchy can already exist in a directed graph or could be induced from a set of node-attributes. It results in a layered style in which vertical positions are assigned first, followed by horizontal ones to reduce edge crossings. A version of this has also been applied by Brandes et al.[93] using the vertical attribute to depict a person's status in a social network and by Brandes and Wagner[51] by clustering and using each cluster as a layer.

A similarly inspired method by Koren and Harel[94] is an application of one-dimensional layout optimisation by axis separation. A one-dimensional layout is computed, in which the layout of nodes on each axis can come from a node-attribute, a particular aesthetic consideration or the application of an existing layout algorithm to one dimension (Koren and Harel give an example of using Kamada and Kawai's), each of which provides a one-dimensional coordinate. The second dimension can then be left to show hierarchy or clustering in the data, another node-attribute or, as the authors propose, a web-based example in which the same nodes emit two different graphs (one showing hyperlink structure and the other page similarity) and produce a one-dimensional layout for each, then combine the axes into a two-dimensional plot.

Mapping nodes to one axis is also the rationale behind Krzywinski et al.'s Hive plots.[95] Multiple axes are positioned radially and nodes are divided between axes according to some attribute or their connectivity structure and then distributed along each axis according to another attribute. An edge is drawn as a curved line between axes and nodes may appear more than once in a layout in order to clarify structure. They call the layout a 'rational visualisation of networks' and a panel of hive plots can be created to view multiple attributes at the same time. The main aim of hive plots is to uncover structure in the graph that could not be previously seen with other network diagrams and to be able to do so for graphs of any size. In Figure 11 a hive plot shows the dependencies between classes in the Flare visualisation tool-kit. Nodes are clustered by class type and the two distinct axes show nodes which are either only source or only target nodes. The duplicated axes show nodes which are both source and target nodes.

Most constraint-based layouts take the form of solving an optimisation problem subject to some constraints. He and Marriott[96] recognised that adding constraints to the layout could aid interactivity through preserving the user's mental map of the layout when nodes are moved. Realising that adherence to graph drawing principles was restricting users' understanding of the underlying semantics of the graph, they proposed three implementations (plus one for trees). The first was a constrained version of Kamada and Kawai's algorithm and the second a polynomial approximation to this constrained version. The third combines the two methods by initially applying the polynomial approximation method to the graph, which gives performance benefits (up to four times faster) followed by the full method, which optimises the layout to better adhere to some of the graph drawing principles.

Much of the work on constraint-based layout methods has been done by Dwyer and various co-collaborators, though initially they focused only on direction as an attribute in DiG-CoLa (constrained layout of digraphs)[97,98] basing it on Sugiyama's method but adding a further optimisation procedure for consideration of the aesthetic principles. They extended this by adding orthogonal ordering constraints, which imply relative node position in relation to all other nodes (a kind of spatial arrangement attribute), again preserving the user's mental map.[99]

A more generic method from Dwyer et al.[100] is IP-Sep CoLa (Incremental Procedure for Separation Constraint Layout of graphs). This method forces a separation between nodes by adding constraints such as organisation into horizontal or vertical layers, containment in a fixed area (as shown in Figure 12), fixing node positions and non-overlapping nodes by

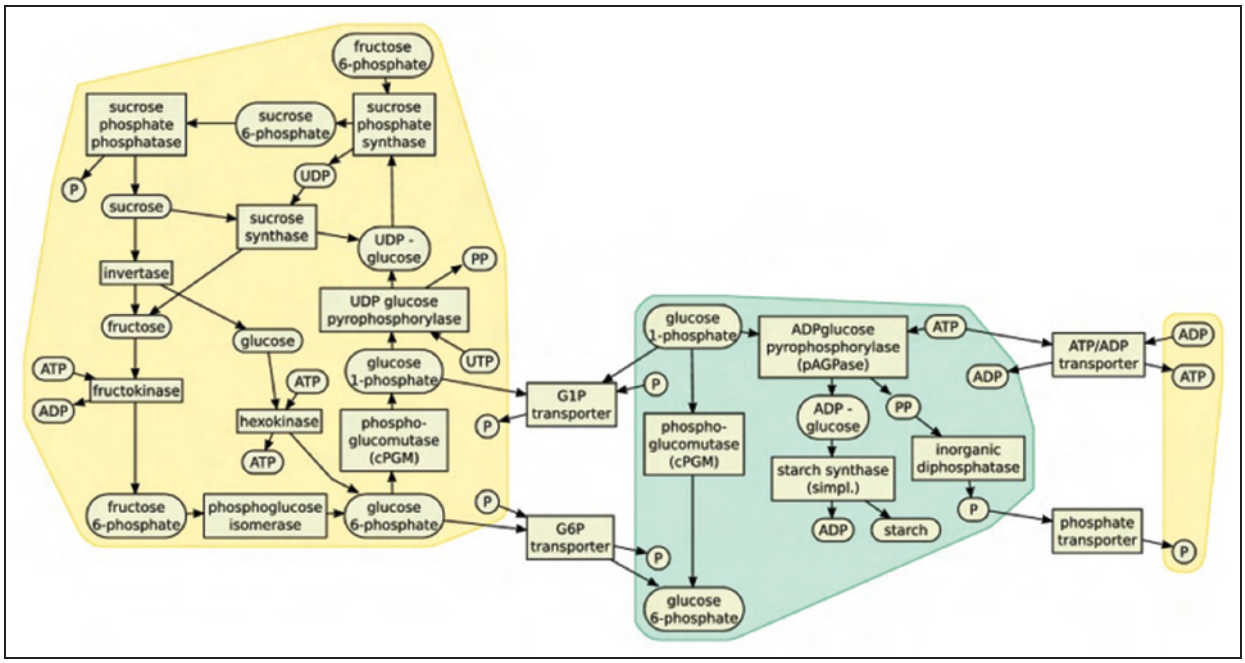

Figure 12. A layout of a metabolic pathway with containment constraints (originally published by BioMed Central in Schreiber et al.[106]).

considering certain attributes. This method was suggested as being useful for displaying protein interaction networks by Barsky et al.[101] and Jianu et al.,[102] but both found that its adaptability is negated by its complexity; however, Dwyer et al.[103] make use of the method interactively by implementing it in the detailed view for overview + detail graph exploration. It is further extended by its use for continuous layout in the diagramming tool 'Dunnart' by preserving topology in response to user interaction.[104,105]

The biology community is a particular driving-force behind the use of node-attribute data for constrained graph layout.[27,101,102] This is often because there are already conventions that depend on node-attributes (from when layouts were drawn manually) that failed to be satisfied by automatic layout algorithms. Examples are Cerebral from Barsky et al.[101] for the inclusion of sub-cellular localisation, which positioned the regions in layers with a modified version of simulated annealing. Jianu et al.[102] and Genc and Dogrusoz[27] required signalling pathway drawing conventions in their interaction network (see Figure 13) and Jourdan and Melancon[107] required the use of conventions from metabolic and regulatory pathways. As Barsky et al.[101] put it, the nodes need to be positioned in a 'biologically meaningful' manner.

MagnetViz[108] also blends force-directed techniques with attributes for layout. Initially the graph is laid out as an adapted version of Tunkelang's[19] force-directed layout from which point the layout become interactive. A user can then assign a virtual magnet to represent a particular attribute and decides where to place this on the graph. Nodes then rearrange their locations depending on the presence of that attribute. Those nodes which have that attribute are attracted towards the magnet and it is up to the user to decide how strongly attracting each magnet can be; the user can also define a boundary inside which all nodes with that attribute should be. Multiple magnets can be used in the layout, which support the use of logical operators for combinations of attributes. After each addition of a magnet the graph reorganises its layout to reflect the new forces in the graph. Problems with the approach mainly lie on the user's side in that the user cannot be relied upon to create layouts that are necessarily the most useful for the the tasks they want to accomplish, may put magnets in inappropriate positions or not select the best combinations of attributes to use. Spritzer and Freitas accept that the tool can have a steep learning curve, but they have also proposed a number of improvements they will make to the tool including limiting the strength of the magnets, changing the boundary size in proportion to the number of nodes included within it, adding filters to hide nodes and incorporating different layouts for nodes attracted to different magnets. However, if the method is correctly applied then it can produce layouts that are 'aesthetically pleasing and semantically relevant' that

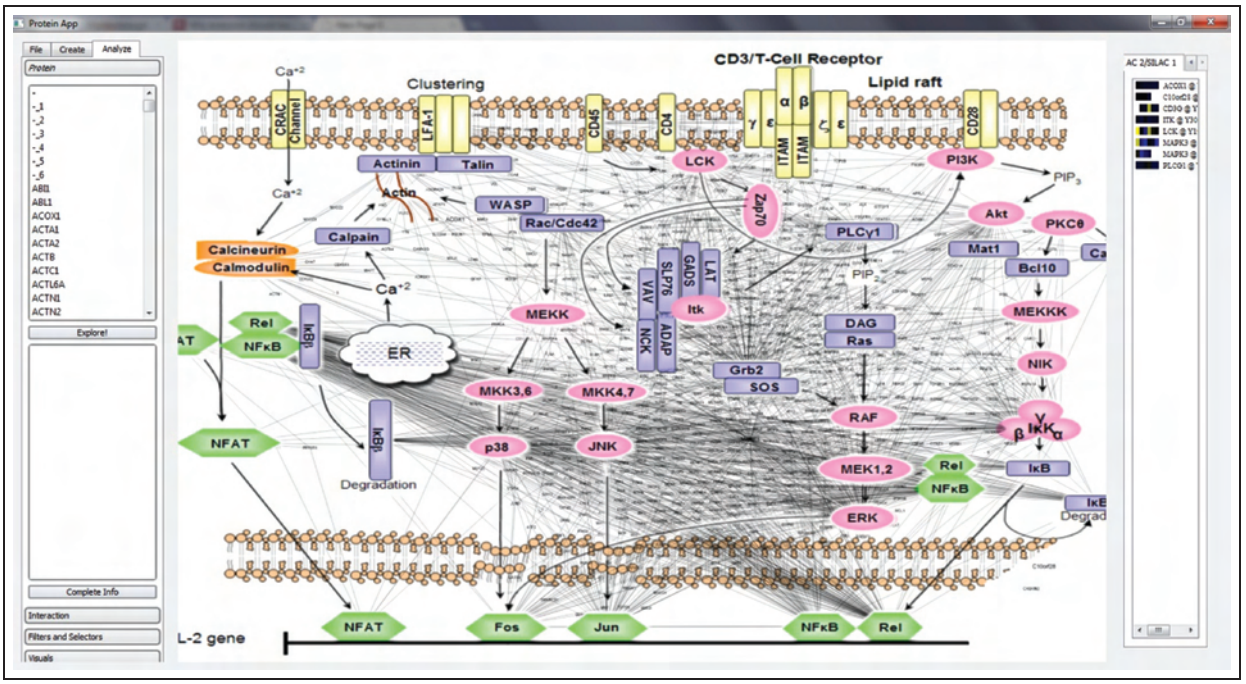

Figure 13. A screenshot of a protein interaction network overlaid on a signalling pathway where the pathway provides a scaffold for the network to be positioned around.[102]

enable the user to complete many of the graph analysis tasks defined by Lee et al.[80]

A pure attribute-only-based generic constraint-based layout technique is Shneiderman and Aris's NVSS (Network Visualisation by Semantic Substrates).[76,109] Nodes are placed in non-overlapping regions according to the value of a particular attribute; nodes inside these regions can then be further positioned with respect to other attributes. Each region is described as a semantic substrate, and proportioning substrate size to the number of nodes it contains allows a quick and easy comparison of category sizes while the use of the regions makes it easier to identify links between categories and to follow connections. It has proved useful in the analysis of legal precedent data, and an example network is shown in Figure 14.

Genetic algorithms provide another way of encoding constraints into graph layout. Like some force-directed methods they do not solve the optimisation problem directly but follow the genetic operations of mating and mutation. The idea is that good genes will thrive and bad genes will die out. In graph layout this is akin to reaching the optimal layout according to the specified constraints.[110] Kosak et al.[111] specified that perceptual organisation (the principles of grouping from Gestalt laws) was more important than layout aesthetics and proceeded to lay out the graph using a genetic algorithm according to conformation to these features (such as clusters or zones, sequences, alignment and symmetry), syntactic validity (overlapping nodes or intersection of nodes and edges) and also some graph drawing principles. The genetic algorithm was more successful than following these rules alone and its strengths lie in being able to handle multiple interacting visual organisational features and aesthetics, while at the same time claiming to produce layouts of excellent quality. Two drawbacks of the method are that the genetic algorithm may never find a valid layout and that the algorithm may take a long time to converge. Branke et al.[112] combined each iteration of a genetic algorithm with the spring embedder until it no longer improved the quality of the graph. They tested it on graphs that are usually difficult for the spring-embedder to lay out successfully alone and found that it was able to produce graphs that reduced the number of edge crossings compared with the force-directed layout.

As some constraint-based layouts include one or more of the graph drawing principles as a constraint, some of these layouts can be analysed in a similar respect to those in 'Family of force-directed and related graph drawing algorithms'. Sugiyama et al.'s layout, for example, considers the principles of edge crossings and keeping adjacent vertices close, and can lay out graphs with more than 500 vertices. Brandes and Wagner built on Sugiyama et al.'s layout to consider the constraint of clustering. Hive plots do not follow the graph drawing principles and once all the

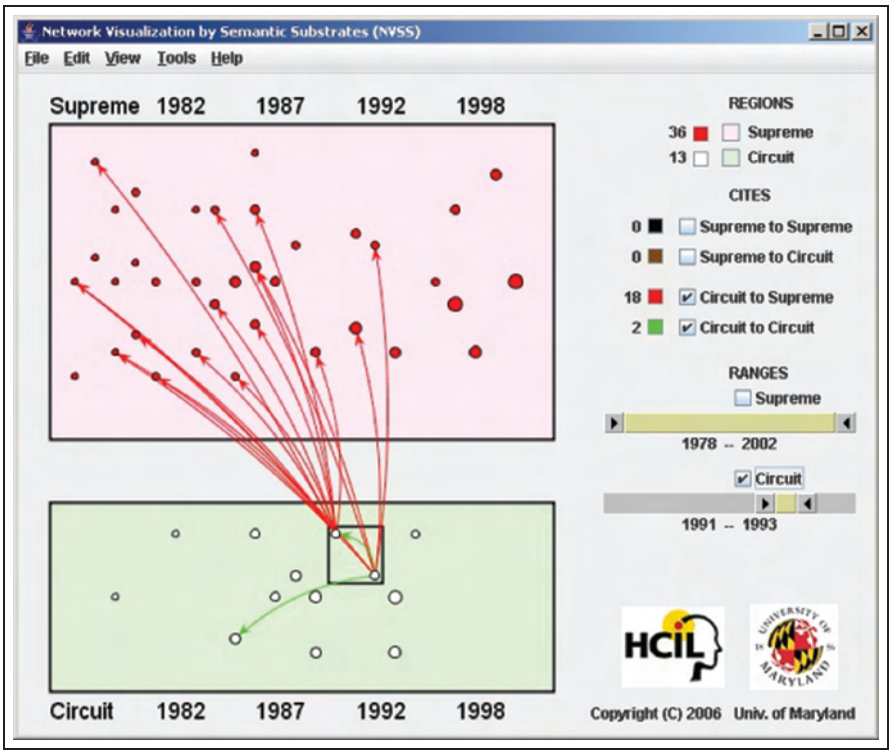

Figure 14. Network visualisation by semantic substrates. The data set visualised is one used for analysing legal precedents where each node represents a citation in either the supreme or circuit courts. Reproduced with the permission of Shneiderman and Aris.[109]

data for the axes are computed it takes less than one second to lay out a graph with many thousands of nodes. In an application-based context hive plots have been compared to the ForceAtlas layout, demonstrating how they are able to more clearly show certain structural features of the graph. The addition of constraints to the Kamada–Kawai method by He and Marriott[96] does not significantly affect running time and they were even able to combine it with a polynomial approximation.

DiG-CoLa[97] was tested against a Sugiyama-style layout and, while it produced more edge crossings for smaller graphs, DiG-CoLa had fewer edge crossings for the larger graphs. Dig-CoLa's edge lengths were also more uniform and it took seven minutes to lay out approximately 2000 nodes. Meanwhile, IP-Sep CoLa also included support for clustering, preventing node–label overlap and various positioning constraints. In 100 seconds IP-Sep CoLa can lay out 10,000 nodes. Barsky et al.'s Cerebral can lay out a few thousand nodes if the number of attributes is kept to around 12 or fewer. They considered edge crossings, node-edge crossings, edge-lengths and also biological proximity in the optimisation of their layout. MagnetViz reported that their method was able to produce an aesthetically pleasing graph whereas NVSS focused on clustering, groups and filtering. The genetic algorithms did not consider specific principles; only that the user could incorporate whichever principle he or she may wish into the optimisation.

Clustering-based attribute layout

Showing clustering in the layout of the graph is one of the most efficient ways of communicating structure in a graph to users. Users even completely neglect the aesthetic of edge crossing in favour of clustering the graph.[79] 'Energy-based approaches' reviewed Noack's[37,38] proposal for a force-directed approach for showing clustering (although the clusters were determined by the algorithm rather than being a defined attribute). The OpenOrd[36] layout mentioned in 'Computational improvements' also groups nodes into clusters according to topology. Methods do exist, however, to either explicitly show clusters already encoded as attributes or to show clusters that have been inferred from attribute similarity.

Extending the semantic substrate idea from 'Constraint-based layouts' is the Group-in-a-Box layout[113] (as shown in Figure 15), which groups each cluster into the rectangle of a treemap and each cluster is then laid out individually inside each rectangle.

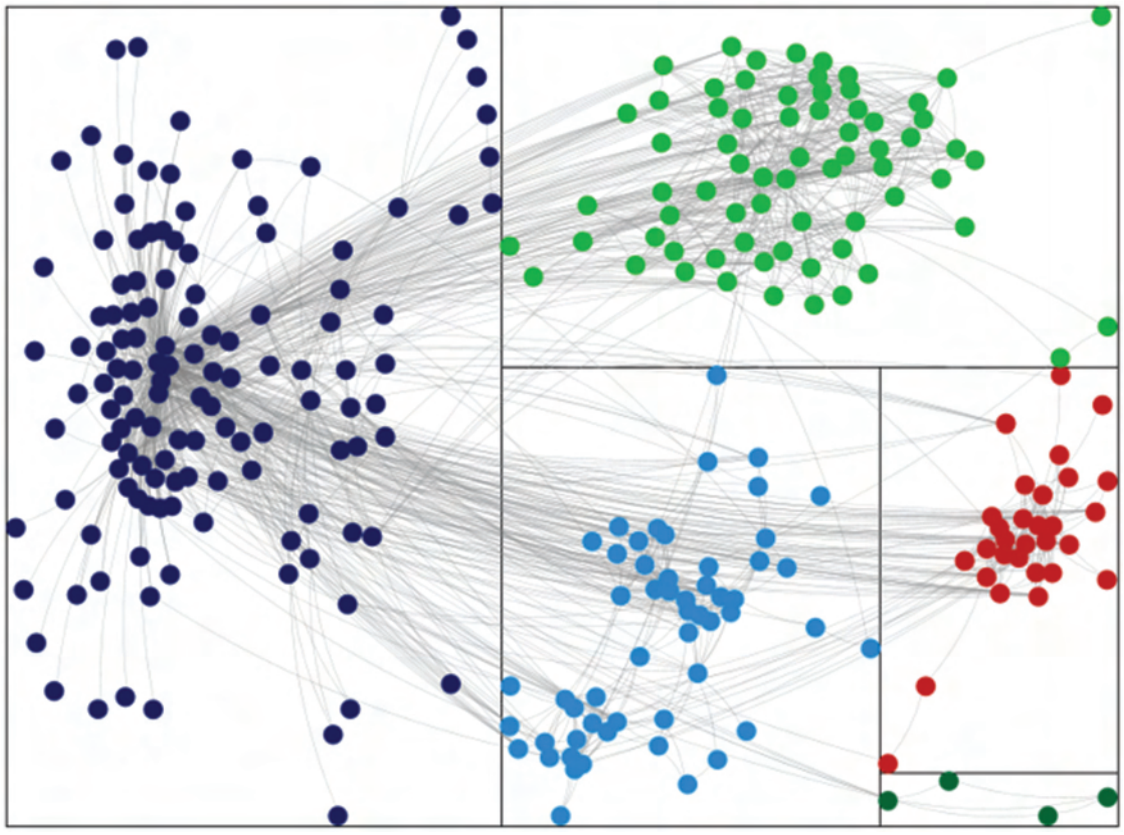

Figure 15. The Group-in-a-Box layout from NodeXL of the clustered protein interaction network.

Other treemap approaches have also been used. Fekete et al.[114] took the idea of decomposing every graph into a tree structure plus some remaining edges and then using this tree structure to display the graph as a treemap upon which the links between nodes can be overlaid, as did Muelder and Ma.[115] However, while Fekete et al. put all nodes that did not fit into the computed hierarchy into a separate section, Muelder and Ma used hierarchical clustering to compute the tree structure, so that all nodes were part of the treemap. Muelder and Ma[115] showed their approach scales to over 300,000 vertices and that it can be used both independently and as an initiation strategy for force-directed layouts.

Following their treemap layout method Muelder and Ma[116] went on to propose the use of a space-filling curve for layout. The nodes are positioned along the curve according to some computed node ordering to maximise the usage of space. For weighted graphs single linkage clustering is used and for unweighted graphs a community structure algorithm is used to decide the node orderings. Muelder and Ma demonstrate that the technique can be used for graphs with over one million nodes. They compare two of their space-filling layouts with a treemap layout[115] and various algorithmic layouts (LinLog, GVA, FM^3, GRIP, ACE and HDE) on a graph with 6000 nodes. While ACE and HDE were clearly the quickest (about 3.5 times faster than the space-filling layouts) their layouts showed no structure at all. GVA, LinLog, GRIP and FM^3 were all slower (1.5 s for GRIP to three minutes for LinLog) and produced hairball-type layouts. The treemap and two space-filling layouts all computed in a similar time of under one second. Their layouts were clearly able to show a clustered structure because a clustering algorithm was used as part of the layout to decide the node ordering in the space-filling case and for the treemap in the other case. This leaves a dependency on being able to detect a clustering in the graph, but if one exists the method is very useful for visualising the overall structure of the network. Itoh et al.[117] also used a similar space-filling approach for multi-category graphs (i.e. many attributes) that computes a clustering hierarchy from similarities between node-attributes. The graph of clusters is then laid out using a simple force-directed method and nodes within each cluster are delineated by a rectangle and laid out using a space-filling algorithm.

Both GOlorize[29] and GraphScape[118] extend force-directed methods to use attributes to show clustering. GOlorize adds a virtual node to the graph to represent each cluster. Virtual edges are then added from the virtual cluster node to members of that cluster. This then drags apart the clusters using Fruchterman and Reingolds's force-directed algorithm, which is very similar to the general method proposed by Huang and Eades.[119] With GraphScape, nodes are clustered based on attribute similarity first and then a modified version of the spring-embedder algorithm is applied to reflect this similarity. Additionally, running time is improved if the FADE[120] algorithm is implemented alongside layout and certain attributes are emphasised through a mountainous landscape metaphor where height represents some attribute value.

Pretorius[121,122] proposes two attribute layouts based on a hierarchical clustering procedure. The first is applied to a large ($>50{,}000$ nodes) state transition graph for which the user initially selects a number of attributes they are interested in an order of importance. The nodes are divided into clusters based on their values for the most important attribute; these clusters are then sub-divided according to their value of the second attribute. This recursive sub-partitioning continues for each attribute and the hierarchy is pictured as the background to the graph. Nodes are placed horizontally along the bottom of the hierarchy according to their attributes. For numerical attributes bar charts show the presence of these attributes below the graph and correspond to the levels in the clustering hierarchy. The transitions between states are then shown as arcs, which indicate direction enabling the user to combine the attributes and the graph to identify recurring patterns in the data.

In the second multi-variate graph visualisation Pretorius[122] again uses the context of a state transition graph. In this case all nodes are placed in a vertical line on both the left (as a source) and right (as a target) sides of the graph in order to indicate direction. As the technique can also visualise edge attributes, labels describing the types of edges are placed between the two sets of nodes. Each edge of that type must pass through its corresponding label's box. The order in which the nodes appear depends on their attributes and, much like the previous method, there is a recursive clustering approach of partitioning based on the first attribute and then on the second, and so on. The method scales to more than 10,000 nodes and users were able to analyse the state transition graphs more effectively. This suggested to the authors that the technique could be applied to multivariate social networks and even be generalisable to all multi-variate graphs.

Mapping attributes directly to 2D space

Perhaps the most obvious use of node-attributes in graph layout is using an attribute that already represents a position. This may be a position such as geo-coordinates, which allow the graph to be superimposed over an image such as a map,[123,124] or placed as they would line up on a soccer[125] or an American football pitch.[88] However, just because a node has a spatial attribute does not mean that it is necessarily the most informative projection of the data; for example, the most interesting patterns might not be geographic ones.[126]

The direct mapping of two attributes to Cartesian coordinates in 2D space is also possible. By way of example, consider the aggregated graph layout, PivotGraph.[127] PivotGraph is based on the idea of pivot tables from spreadsheets and uses an OLAP (Online Analytical Processing) database model in order to produce a grid-based graph showing two categorical attributes with node sizes representing the number of nodes with that attribute. This technique allows the exploration of the relationships between attributes in the graph through collapsing and expanding the attribute nodes; however, it can also obscure the topology of the graph by making graphs appear connected or cyclical when they are not.

Two similar methods for SNA (Social Network Analysis) are given by Bezerianos et al.[13] and Viau et al.[128] Bezerianos et al.'s GraphDice builds on a similar technique for scatterplots called ScatterDice[129] and aims to be more of an exploratory tool than other SNA visualisation tools, which tend to be confirmatory. Like Viau et al.,[128] it provides two views of the data. One is an overview scatterplot matrix (seen on the left in Figure 16), which shows a small multiple of each possible combination of attributes, while the other larger graph view (seen to the right) shows a full image of the graph based on the two currently selected attributes. GraphDice uses attributes that are application specific (e.g. in a co-authorship network, papers written, citation count, field) as well as node centrality metrics. Viau et al.[128] also allow parts of the graph to be laid out manually or using a force-directed layout.

GraphDice received positive feedback from a user working in the history domain while Viau et al.'s implementation also received praise from the biologists with whom they were working, who claimed it could prove to be a valuable part of their work flow.

There is no limit to the number of attributes a node can have. Therefore, it should be possible to use all of these attributes for layout. A number of techniques using dimension reduction have been proposed, the first of which was a multi-dimensional visualisation of state transition graphs by Pretorius and van Wijk,[130] in

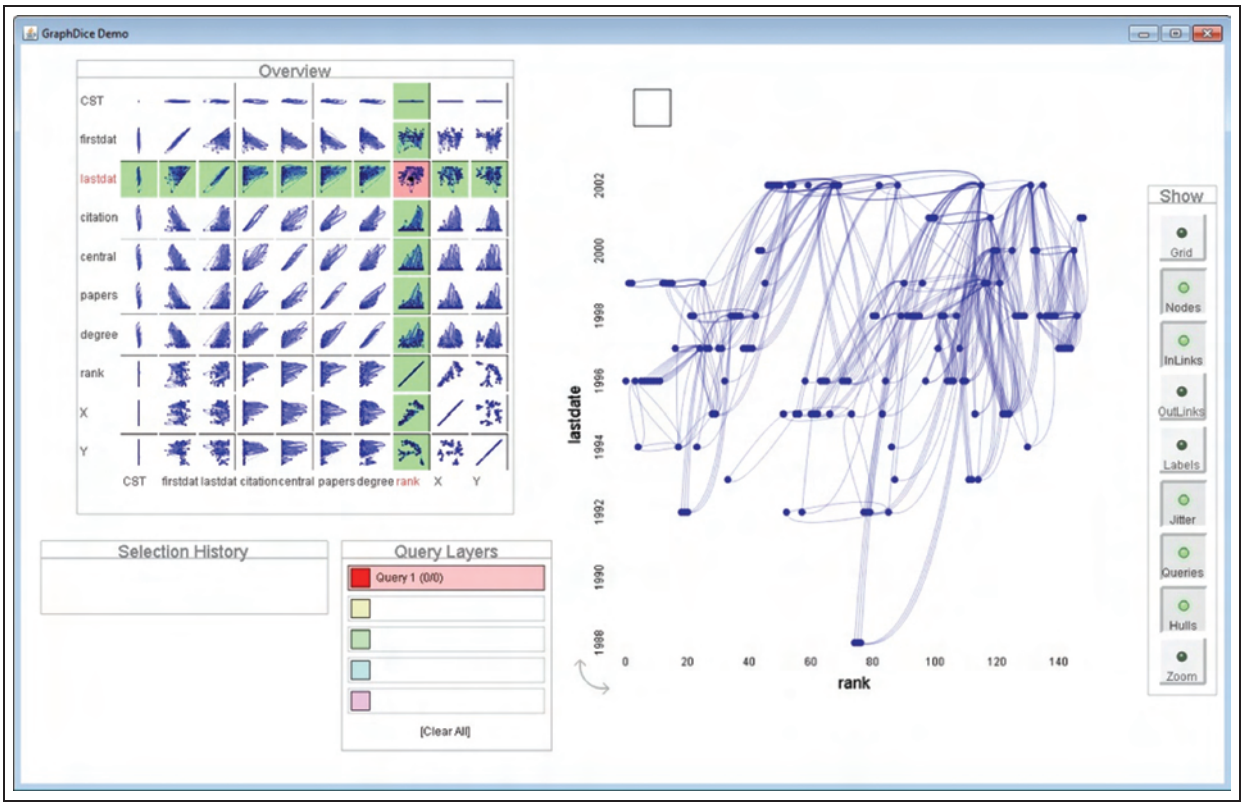

Figure 16. GraphDice.[13] A screenshot of the GraphDice tool showing a graph using the InfoVis 2004 contest data set, in which nodes are positioned based on rank and the last date of the conference.

which the graph is composed of the state transitions and the attributes are the variables of each state. A user then selects a subset of variables to be included in the visualisation and five possible projections into two dimensions are offered: uniform distribution, manual distribution, hypergrids, rotated hypergrids and a PCA projection. However, ultimately for this application they found that simply visualising the state variables as parallel histograms enabled them to fulfil the tasks they wished to carry out with the graph most successfully. The PCA projection allowed them to see the different phases of the system easily, but they were unable to find correlations between the graph and variables.

Since Pretorius and van Wijk's attempt there have been more successful efforts at using dimension reduction for visualisation of graphs using node-attributes. GeoSOM[131] (geodesic self-organising map) was a hybrid approach to laying out graphs using both node-attributes and properties of the graph based on a world metaphor. The SOM is trained to place vertices with similar attributes close to one another and the algorithm was improved in order to non-linearly take into account the graph distances between vertices (the same approach as Kamada and Kawai's[23] force-directed method), resulting in a layout that reflects both the graph's structure and its attributes. Evaluating their layout, they found that the non-linear incorporation of graph distance rather than linear[132] decreased the number of edge crossings in the layout. From a user study they also found that users were able to combine the structure and the attributes to extract useful information from the graph better than they were able to using a force-directed layout and a glyph approach for the presentation of the attributes. An example of the GeoSOM layout for an international metal trading network is shown in Figure 17.

Two recent methods that also use the idea of dimension reduction are EdgeMaps by Dork et al.[133] and Gibson and Faith's[134] application of targeted projection pursuit (TPP) to graph layout. The aim of EdgeMaps was to unite the visualisation of node–link diagrams that show the explicit relationships between nodes in the graph and multi-dimensional scaling techniques used to visualise implicit relationships, i.e. attribute data. The layout is produced from an MDS projection of attributes onto a Cartesian space and position is double encoded by hue and saturation. In particular, three application areas are explored: influence relationships of philosophers, musicians and artists on others in their field. Implicit relationships are

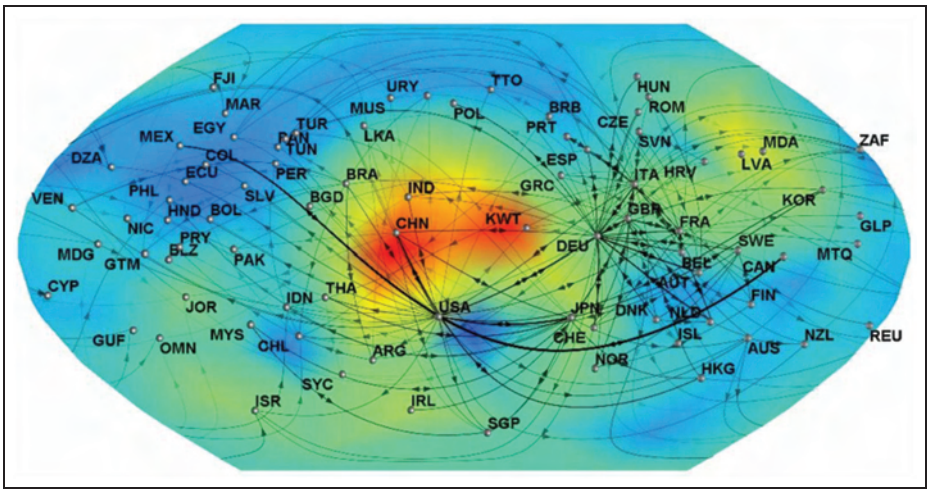

Figure 17. The GeoSOM layout for an international metal trading network (source: http://rp-www.cs.usyd.edu.au/~chwu/CerealMetal.htm).

those such as interests, musical genres or artistic movements. Only one node's links are shown at a time for readability, which means that the user can explore only one set of influences at a time and cannot, for example, easily explore influence similarity between two nodes. In addition there is no indication as to which attributes have resulted in two philosophers being placed close to each other. An example of the projection is shown in Figure 18(a) and the influence edges in Figure 18(b).

Targeted projection pursuit for graphs[134] also considers multiple node-attributes and relates each one to a dimension. Initially, nodes are laid out using a PCA projection based on the node-attributes. From this the users can begin to explore the graph. A user can grab any set of nodes and try to reposition them to reflect some intuitive idea or hypothesis they have about the data. The nodes will move to this position only if there exists some projection from the original data space to two dimensions that results in a view equal to the target view, i.e. the view desired by the user. Otherwise the projection that is the nearest match to the target view is found. The technique is particularly useful if the user has some clustering in mind for the data; he or she can try to separate each of the clusters while observing the pattern of edges between those clusters and determining whether the attribute data follow the same pattern as those clusters or another pattern altogether. This separation can also be automated to find the maximum separation between clusters. The user then can additionally assess which attributes are most significant in the chosen layout aiding the feedback loop between the graph's topological structure and the attributes of the nodes. Figure 19 shows an image of how the graph can be clustered according to its attributes. Although the technique shows promise for future applications so far, its usefulness has been demonstrated on only a small synthetic data set replicating a small-world graph.

Discussion of the use of node-attributes in layout

Clearly there is some potential in using node-attributes for layout. As more data are collected, understanding how they combine to form a whole is an important task. For example, Bezerianos et al.[13] state that nodes in social networks can have as many as 30 attributes; knowing how these attributes affect or even govern the interactions in a social network can have a significant impact on how it is analysed or the insight gained from it. Other advocates of this idea have also identified its potential to create a deeper understanding of the graph as a whole including the potential correlations between graph topology and the underlying attributes.[86]

There do not have to be as many as 30 node-attributes to be able to use them for layout. Graphs which have only a few attributes are prime candidates for graph visualisation through one of the constraint- or clustering-based methods. Hive plots, for example, require only one attribute for their layout and the user can then define how they want to partition that attribute to form the axes of the layout. They are useful for focusing on one attribute and its relationship to the graph's structure. Being able to understand and analyse their output requires a steep learning curve; however, they are able to visualise large graphs with many thousands of nodes and provide a structural perspective that other layouts do not. In biology, utilising a few attributes for layout can prove extremely fruitful

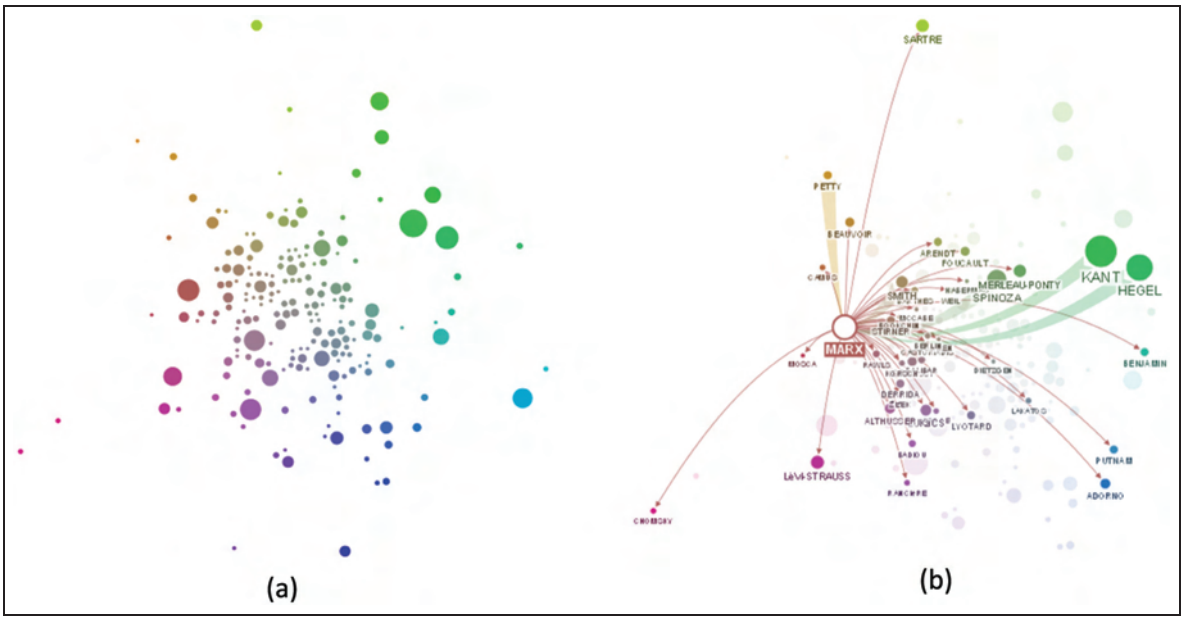

Figure 18. The EdgeMaps[133] graph visualisation of philosopher's influence relationships (images created from http://mariandoerk.de/edgemaps/demo/). (a) is the initial layout presented to the user and (b) the edges shown when Marx is selected.

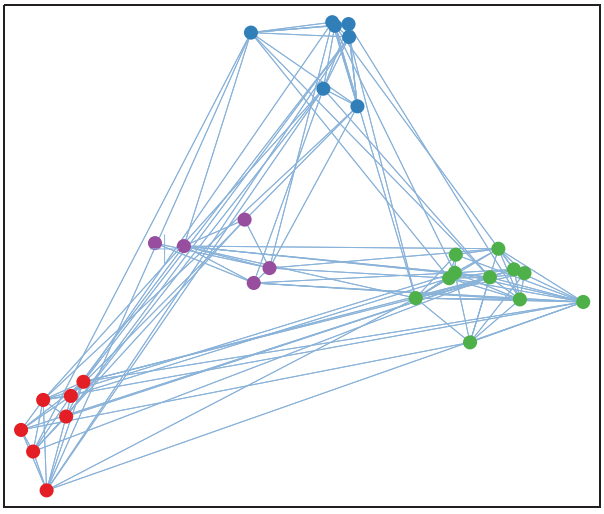

Figure 19. The targeted projection pursuit layout for graphs (from ref. 134).

and exactly what users want. One such case was Barsky et al.'s[101] layout, which provided a clear way to separate proteins in a cell and a visual indication of which part of the cell they exist in. However, even this technique quickly runs into problems where there are parts of the cell are contained inside other parts, e.g. mitochondria inside the cytoplasm.

MagnetViz gives users a lot of control over the layout and by combining it with a force-based layout some of the topological structure of the graph remains in the layout. The interaction and observing how the graph rearranges itself as magnets are added or taken away may also improve the user's understanding of the relationship between the attributes and the graph's structure. NVSS allows multiple types of nodes, each with different attributes, to appear in the same layout, which means that there are even fewer restrictions on the data that can be included in the graph, though the technique is difficult to scale to larger graphs without interactive filtering.

When manually drawing graphs users tend to neglect aesthetic criteria in favour of displaying the clustering of the graph;[79] thus, layouts that automatically cluster the nodes according to some predefined criteria have a good chance of being useful to the user and are an important structural feature for understanding the graph. Layouts that show clustering range from the Group-in-a-Box layout, which delineates each cluster into a box that can then be laid out, Muelder and Ma's treemap and space-filling approaches, which scale to hundreds of thousands of nodes and give that all-important strong overview of the structure of the graph.

Sometimes using just a few attributes is not enough. To begin with, a user might be visualising the graph to gain a better understanding of the data in it and may not know which attributes should be focused on – an issue identified by Shneiderman and Aris[109] in NVSS. In that case, moving between scatterplot matrices, such as in GraphDice, provides one way of having

multiple attributes for layout at a user's fingertips. However, the user can still investigate a relationship between only two attributes at a time, and too many attributes would make the scatterplot matrix unreadable and negate its utility.

GeoSOM, EdgeMaps and TPP are three methods which try to overcome this limit on the number of attributes by utilising dimension reduction techniques to project the graph onto a 2D space. They all consider multiple attributes and, therefore, multiple dimensions at once, but all three also leave room for improvement. GeoSOM is the only method that takes into account topological structure when computing the layout as well as the attributes, but using the world metaphor means it is difficult to comprehend edges that leave the graph on one side of the world and reappear on the other. Also from Figure 17, it can be seen that if the size of graph were to increase any more it would become quite difficult to read.

EdgeMaps restricts the display of links at any one time and, while this improves readability, it also results in a loss of context if a user wishes to compare two or more nodes or to look for a similarity between attribute structure and topological structure. Even simply allowing multiple selections of nodes to be in focus would, in part, solve this problem and at least allow the user to explore edge patterns between nodes. Although attributes are used in the MDS projection the user then has no idea how the attributes have influenced the positions of the nodes in the final projection, so some of the insights that could have been gained from this layout are lost.

TPP, on the other hand, does show all of the links in the graph and allows the user to interact directly with the graph to find a layout and form hypotheses based on how influential certain attributes are in the current projection to provide an automatic way to determine relationships between node-attributes and cluster membership. This method now needs to be validated on real-world data with many more nodes and edges in order to assess its effectiveness as a network visualisation technique.

Graph layout that combines node-attributes and dimension reduction provides one solution for producing an overview layout with meaningful positions for nodes that can communicate a structure of the graph related to specific properties of the nodes. This sort of overview is particularly suited to tasks of exploring the data, finding clusters and identifying patterns and outliers.[80] Using node-attributes for layout should improve this insight that can be gained from the graph by allowing the user to make visual inference from the links between nodes, their positions in space and the combination of the two. In almost all these cases, though, there needs to be more evidence of application to specific use cases, extensions to larger datasets, evaluation against other methods and specific criteria as to when these methods are best applied.

Discussion

Over the past 30 years many solutions to the problem 'What is the best way to draw a graph?' have been put forward. Initially, most of these algorithms were force-directed approaches and these have remained consistently popular over this time. As such, they have found themselves integrated into many graph visualisation tools, thereby reinforcing their popularity. These algorithms gained acceptance by drawing a graph that is said to conform to several graph drawing aesthetic principles. The drawing of graphs with these algorithms often suffers from high running times and the production of a non-optimal graph layout. Beyond that there is also a problem with validation of the use of these principles for layout in terms of aiding users' understanding and enabling them to extract meaning from the graph. Studies assessing the impact these principles have on layout tend to be limited to a small set of nodes (fewer than 20) and a limited set of tasks that are not necessarily applicable to a much larger graph. Even Huang and Eades[135] in their eye tracking study said that they did not know if their results would scale with the size and complexity of the graph.

Force-directed methods seem to be optimised for a particular type of task, such as those indicated by Purchase[24] and Lee et al.[80] (e.g. counting node degrees and identifying adjacent nodes and further connections). However, both Dunne and Shneiderman[17] and Salvini et al.[136] comment on the lack of knowledge about the relationship between drawing principles, layout and task.

One effort to overcome this lack of understanding is Purchase's[24] study on how the principles, such as symmetry and edge crossings, and layout are related to the ability to complete three tasks. These tasks were finding the shortest path between two nodes and identifying which nodes, and, separately, edges, that would, if removed, disconnect part of a graph. She found that for the three force-directed graph layouts studied little variation was found in the time taken to complete the tasks or in the errors made when answering with respect to each different layout method. This indicates that where graph size is small, there is no reason to choose one layout over another, at least in the case of performing those three tasks.

When looking at this study the first thing to consider is how appropriate the tasks were for analysing layout. Were the tasks chosen representative of tasks where it is essential to have a good layout to complete

or are there other tasks on which layout has more impact? The second is whether the principles used to create these layouts are actually successful in aiding users to carry out this type of task. Third, are the results the same for much larger graphs or is there a more perceptible difference on task completion and are the differences between the layouts more apparent?

Graph size in general is a very important consideration when analysing the success of the effect of layout. Newer layout algorithms developed have tried to improve upon the results given by force-directed layouts; these included multi-level techniques that make force-directed approaches more efficient for graphs of larger size and the use of dimension reduction techniques, which make use of graph-theoretic distance for computing node position, resulting in layouts similar to force-directed ones. These techniques still tend to keep the guiding graph drawing principles as a goal and because of this are likely to produce good layouts for only sparsely connected graphs or those which form a mesh or grid-like structure.[50] Except in these cases, they are unlikely to produce layouts that are a sufficient improvement on the force-directed ones.

What comes from this is the question of how do the principles apply in terms of graph sizes? Force-directed methods are still a very good option for graph layout when the graph is small and it is true that for small graphs these principles are successful in computing a comprehensible layout. But it is not understood whether they are as applicable to producing a good layout for larger graphs. For example, Henry and Fekete[8] make the point that finding a good overview is challenge for large graphs and crucial to the following exploration process. This overview should then be connected to a layout which gives a good representation of the high-level structure of a graph placing more emphasis on identifying and analysing spatial groupings, clustering, patterns and outliers but a good overview does not necessarily translate to one that conforms well to the principles of graph drawing.

More successful recent force-directed based techniques have actually been the ones that have ignored certain principles to show off other structural properties of the graph such as the LinLog, ForceAtlas and OpenOrd layouts. All three are still based on the idea of a physical system but the principle they have tried to optimise is one of clustering rather than than being concerned with edge lengths or uniform node distributions, for example. Clustering is only one potential new principle and there may be others that the layout can be optimised to show. For example, decreasing the number of edge crossings is usually an aim for layout but in parallel coordinate plots they are used to highlight an inverse relationship between two dimensions; thus, rather than attempting to minimise the number of edge crossings perhaps they could also be used to emphasise a particular structural feature or relationship. Additionally, with the adaptation of force-directed techniques for clustering it shows that using a physical system model can still be a very strong option for layout if applied in the right way.

Given that there has been such a large body of research on the simulation of physical systems for graph layout and the optimisation of force-directed algorithms, this work should be utilised. Thus, by identifying and then changing the emphasis on which properties of the graph the layout should reflect, force-directed methods could be redesigned to highlight some of these other features of the graph. This should be done in conjunction with an investigation into alternative graph drawing principles and whether they can be principles that can be applied generally, to graphs with certain topological structure or for specific graph sizes. An extension of this would be if to investigate if there is a relationship between the principles used and the type of task the user wants carry out with the visualisation or that can be best solved using a particular layout style.

Even some of the studies on smaller graphs have recognised the importance of presenting a good structural overview. Van Ham and Rogowitz[79] observed users preferring to cluster nodes in their layouts of a graph; most participants in an experiment by Dwyer et al.[137] selected graphs that showed a clique; and semantic grouping and considerations of semantics in general have been considered important for improving user understanding of graph visualisations in information visualisation contexts.[138,139]

The lack of semantic considerations has also been highlighted by users, with many communicating their dissatisfaction at the inability of force-directed techniques to produce a comprehensible layout either because it resembles a 'giant hairball' or because it lacks any context. This is particularly true of users in the biology community who have already expressed their desire, and made some attempts, to have a graph layout that is meaningful and improves their understanding.[33,101,106,140]

This includes layout methods making use of attributes. Typically, there is a lot of information available on the nodes, more than just how they are connected to other nodes. Developing computational methods using this philosophy of laying out graphs according to their attributes has potential, with the correct algorithm, to assess the correlations, contradictions and trade-offs between the attributes and focus them into a comprehensible, informative layout.

Layouts that require the incorporation of node-attributes are becoming more frequent and are able to give users a context they cannot get from purely

algorithmic methods, no matter how good the layout might be. Layouts with attributes also range from the very simple, showing only one additional attribute (such as a cluster or the separation of axes based on numerical values that can be achieved using hive plots), to dimension reduction methods in which there is no theoretical limit on the number of dimensions that can be incorporated into the layout. They provide a connection between the graph and all the other data that are potentially available, and allow the user to not only answer questions (e.g. whether A and B are connected) but also to determine if there is something about their attributes that is able tell them why they are connected.

However, node-attribute-based layouts do not escape criticism and leave plenty of room for improvement. A major drawback of some of these methods so far is their lack of demonstrated scalability to larger datasets. Those layouts, such as EdgeMaps and TPP, can visualise high numbers of attributes but they also need to show that they can effectively use those attributes to visualise high numbers of nodes and edges. In addition to aiding the exploration of the relationship between the topological structure and the attributes they should also facilitate the completion of tasks related to the graph's structure.

There are three other potential methods for improving node-attribute layouts further. The first is to follow the promising leads of GeoSOM and MagnetViz so that both the graph's topological structure and its attributes influence the layout; a pure node-attribute layout might miss some interesting topological features. This can even be by adding graph theoretical distances between nodes as additional dimensions for the dimension reduction techniques. Dimension reduction is a well-studied area and so a further enhancement may be to utilise some of the state-of-the-art dimension reduction techniques for visualisation and use them for layout. Finally, it has also been suggested that interaction, or at least user involvement, in the layout process, as found with MagnetViz and TPP, is a promising approach to creating new layouts.[141]

This review has presented some of the most commonly used and implemented techniques for graph layout in information visualisation. This was followed by a discussion of the current state of graph layout, which emphasised in particular those techniques that make use of node-attributes for layout. It has discussed some of the merits and deficiencies in both of these approaches to graph layout and has suggested some directions for further research, particularly in regard to understanding what users want from a layout, which tasks are to be completed with the layout and how the size of the graph to be laid out affects the layout and the actions of the user. In addition, it has proposed that, although a good start has been made on incorporating attributes for layout, in order to enrich the graph's layout further research should do more to both validate and extend these techniques.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

References

1. McGrath C, Blythe J and Krackhardt D. Seeing groups in graph layouts. *Connections* 1996; 19: 22–29.
2. Tutte WT. Convex representations of graphs. *P Lond Math Soc* 1960; s3–10: 304–320.
3. Tutte WT. How to draw a graph. *P Lond Math Soc* 1963; s3–13: 743–767.
4. Blythe J, McGrath C and Krackhardt D. The effect of graph layout on inference from social network data. In: *Proceedings of the symposium on graph drawing, GD '95*, Passau, Germany, 20–22 September 1995, pp. 40–51. London, UK: Springer-Verlag.
5. Wong PC, Foote H, Mackey P, et al. A space-filling visualization technique for multivariate small-world graphs. *IEEE T Vis Comput Gr* 2011, 18(5): 797–809.
6. Graph Drawing. http://www.graphdrawing.org/ (1990, accessed 25 October 2011).
7. Di Battista G, Eades P, Tamassia R, et al. *Graph drawing: algorithms for the visualization of graphs*. Prentice-Hall, Upper Saddle River, New Jersey, USA, 1999.
8. Henry N and Fekete JD. MatrixExplorer: a dual-representation system to explore social networks. *IEEE T Vis Comput Gr* 2006; 12: 677–684.
9. Teyseyre AR and Campo MR. An overview of 3D software visualization. *IEEE T Vis Comput Gr* 2009; 15: 87–105.
10. Herman I, Melancon G and Marshall MS. Graph visualization and navigation in information visualization: a survey. *IEEE T Vis Comput Gr* 2000; 6: 24–43.
11. Knuth DE. Computer-drawn flowcharts. *Commun ACM* 1963; 6: 555–563.
12. Wasserman S and Faust K. *Social network analysis: methods and applications*. Cambridge University Press: Cambridge, 1994.
13. Bezerianos A, Chevalier F, Dragicevic P, et al. GraphDice: a system for exploring multivariate social networks. Computer Graphics Forum, 2010; 29: 863–872.
14. Perer A and Shneiderman B. *Integrating statistics and visualization*. ACM Press: New York 2008, 265 pp.
15. Quinn NR and Breuer MA. A forced directed component placement procedure for printed circuit boards. *IEEE T Circuits Syst* 1979; 26: 377–388.
16. Eades P. A heuristic for graph drawing. *Congressus Numerantium* 1984; 42: 149–160.
17. Dunne C and Shneiderman B. *Improving graph drawing readability by incorporating readability metrics: a software tool for network analysts*. Technical Report no.

HCIL-2009-13, 2009, http://www.cs.umd.edu/localphp/hcil/tech-reports-search.php?number=2009-13
18. Di Battista G, Eades P, Tamassia R, et al. Algorithms for drawing graphs: an annotated bibliography. *Comp Geom-Theor Appl* 1994; 4: 235–282.
19. Tunkelang D. *A practical approach to drawing undirected graphs*. Carnegie Mellon University, Pittsburgh, PA, USA. 1994.
20. Van Ham F and Van Wijk J. Interactive visualization of small world graphs. In: *IEEE symposium on information visualization*, Austin, TX, 10–12 October 2004, pp. 199–206, IEEE.
21. Huang W. Exploring the relative importance of number of edge crossings and size of crossing angles: a quantitative perspective. *Int J Adv Int* 2011; 3: 25–42.
22. Bennett C, Ryall J, Spalteholz L, et al. The aesthetics of graph visualization. In: *Proceedings of computational aesthetics in graphics, visualization, and imaging*, Banff, Alberta, Canada, 20–22 June 2007, vol. 5, pp. 1–8, Eurographics Association.
23. Kamada T and Kawai S. An algorithm for drawing general undirected graphs. *Inform Process Lett* 1989; 31: 7–15.
24. Purchase HC. Effective information visualisation: a study of graph drawing aesthetics and algorithms. *Interact Comput* 2000; 13: 147–162.
25. Wills GJ. Nicheworks-interactive visualization of very large graphs. *J Comput Graph Stat* 1999; 8: 190–212.
26. Fruchterman TM and Reingold EM. Graph drawing by force-directed placement. *Software Pract Exper* 1991; 21: 1129–1164.
27. Genc B and Dogrusoz U. A layout algorithm for signaling pathways. *Inform Sciences* 2006; 176: 135–149.
28. Demir E, Babur O, Dogrusoz U, et al. PATIKA: an integrated visual environment for collaborative construction and analysis of cellular pathways. *Bioinformatics* 2002; 18: 996–1003.
29. Garcia O, Saveanu C, Cline M, et al. GOlorize: a cytoscape plug-in for network visualization with gene ontology-based layout and colouring. *Bioinformatics* 2007; 23: 394–396.
30. Frick A, Ludwig A and Mehldau H. A fast adaptive layout algorithm for undirected graphs. In: Tamassia R and Tollis I (eds) *Graph drawing, lecture notes in computer science*, vol. 894. Berlin/Heidelberg: Springer, 1995, pp. 388–403.
31. Davidson R and Harel D. Drawing graphs nicely using simulated annealing. *ACM T Graphic* 1996; 15: 301–331.
32. Kohler J, Baumbach J, Taubert J, et al. Graph-based analysis and visualization of experimental results with ONDEX. *Bioinformatics* 2006; 22: 1383–1390.
33. Li W and Kurata H. A grid layout algorithm for automatic drawing of biochemical networks. *Bioinformatics* 2005; 21: 2036–2042.
34. Benson D and Adler G. jGraph 2002. http://www.jgraph.com/ (2002, accessed 10 November 2011).
35. Shannon P, Markiel A, Ozier O, et al. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res* 2003; 13: 2498–2504.
36. Martin S, Brown WM, Klavans R, et al. OpenOrd: an open-source toolbox for large graph layout. In: *Proceedings of SPIE*, San Francisco, CA, United States, 23–27 January 2011, vol. 7868, pp. 786–806, SPIE.
37. Noack A. An energy model for visual graph clustering. In: Liotta G (ed.) *Graph drawing, lecture notes in computer science*, vol. 2912. Berlin/Heidelberg: Springer, 2003, pp. 425–436.
38. Noack A. Energy models for graph clustering. *J Graph Algorithm Appl* 2007; 11: 453–480.
39. Jacomy M, Heymann S, Venturini T, et al. ForceAtlas2, A graph layout algorithm for handy network visualization. Pre-print 2011; 1–21. http://www.medialab.sciences-po.fr/publications/Jacomy_Heymann_Venturini-Force_Atlas2.pdf
40. Bastian M, Heymann S and Jacomy M. Gephi: an open source software for exploring and manipulating networks. In: *Third international AAAI conference on weblogs and social media*, San Jose, California, 17-20 May 2009, pp. 361–362, Association for the Advancement of Artificial Intelligence (AAAI).
41. Noack A. *Unified quality measures for clusterings, layouts, and orderings of graphs, and their application as software design criteria*. PhD Thesis, Brandenburg University of Technology, Germany, 2007.
42. Barnes J and Hut P. A hierarchical $O(N\log N)$ force-calculation algorithm. *Nature* 1986; 324: 446–449.
43. Freeman LC. Graphical techniques for exploring social network data. In: Carrington PJ, Scott J and Wasserman S (eds) *Models and methods in social network analysis*. Cambridge University Press: Cambridge, 2005, pp. 249–269.
44. Gansner ER, Koren Y and North S. Graph drawing by stress majorization. In: Pach J (ed.) *Graph drawing, lecture notes in computer science*, vol. 3383. Berlin/Heidelberg: Springer, 2004, pp. 239–250.
45. Kruskal JB and Seery JB. Designing network diagrams. In: *Proceedings first general conference on social graphics*, Washington, D.C., July 1980, pp. 22–50, U. S. Department of the Census. Bell Laboratories.
46. Buja A, Swayne DF, Littman ML, et al. Data visualization with multidimensional scaling. *J Comput Graph Stat* 2008; 17: 444–472.
47. Brandes U and Pich C. Eigensolver methods for progressive multidimensional scaling of large data. In: Kaufmann M and Wagner D (eds) *Graph drawing, lecture notes in computer science*, vol. 4372. Berlin/Heidelberg: Springer, 2007, pp. 42–53.
48. Brandes U and Pich C. An experimental study on distance-based graph drawing. In: Tollis I and Patrignani M (eds) *Graph drawing, lecture notes in computer science*, vol. 5417. Berlin/Heidelberg: Springer, 2009, pp. 218–229.
49. Harel D and Koren Y. Graph drawing by high-dimensional embedding. *J Graph Algorithm Appl* 2004; 8: 207–219.
50. Archambault D, Munzner T and Auber D. Topolayout: multilevel graph layout by topological features. *IEEE T Vis Comput Gr* 2007; 13: 305–317.
51. Brandes U and Wagner D. Visone: analysis and visualization of social networks. In: Junger M and Mutzel P (eds) *Graph drawing software*. Springer-Verlag: Berlin, 2004, pp. 321–340.

52. Hachul S and Jünger M. Large-graph layout algorithms at work: an experimental study. *J Graph Algorithm Appl* 2007; 11: 345–369.
53. Koren Y. Graph drawing by subspace optimization. In: *Proceedings of the 6th joint eurographics – IEEE TVCG symposium on visualization*, Konstanz, Germany, 19–21 May 2004, pp. 65–74, Eurographics Association.
54. Civril A, Magdon-Ismail M and Bocek-Rivele E. SDE: graph drawing using spectral distance embedding. In: Healy P and Nikolov N (eds) *Graph drawing, lecture notes in computer science*, vol. 3843. Berlin/Heidelberg: Springer, 2006, pp. 512–513.
55. Civril A, Magdon-Ismail M and Bocek-Rivele E. SSDE: fast graph drawing using sampled spectral distance embedding. In: *Graph drawing, lecture notes in computer science*, vol. 4372. Berlin/Heidelberg: Springer, 2007, pp. 30–41.
56. Bonabeau E. Self-organizing maps for drawing large graphs. *Inform Process Lett* 1998; 67: 177–184.
57. Bonabeau E. Graph multidimensional scaling with self-organizing maps. *Inform Sciences* 2002; 143: 159–180.
58. Meyer B. Competitive learning of network diagram layout. In: *Proceedings. 1998 IEEE symposium on visual languages*, Nova Scotia, Canada, 1–4 September, 1998, pp. 56–63, IEEE Press.
59. Hendrickson B and Leland R. A multi-level algorithm for partitioning graphs. In: *Proceedings of the IEEE/ACM conference on supercomputing*, San Diego, CA, United States, 4–8 December 1995, 28 pp, ACM.
60. Gajer P, Goodrich M and Kobourov SG. A multi-dimensional approach to force-directed layouts of large graphs. In: Marks J (ed.) *Graph drawing, lecture notes in computer science*, vol. 1984. Berlin/Heidelberg: Springer, 2001, pp. 211–221.
61. Hu Y. Efficient, high-quality force-directed graph drawing. *Math J* 2005; 10: 37–71.
62. Hachul S and Jünger M. Drawing large graphs with a potential-field-based multilevel algorithm. In: Pach J (ed.) *Graph drawing, lecture notes in computer science*, vol. 3383. Berlin/Heidelberg: Springer, 2005, pp. 285–295.
63. Harel D and Koren Y. A fast multi-scale method for drawing large graphs. In: *Proceedings of the working conference on advanced visual interfaces, AVI '00*, Palermo, Italy, 23-26 May 2000, pp. 282–285, ACM Press.
64. Walshaw C. A multilevel algorithm for force-directed graph drawing. Lecture Notes in Computer Science. In: Joe M (ed) *Graph Drawing*, vol. 1984: 31–55, 2001, Springer Berlin/Heidelberg.
65. Cohen JD. Drawing graphs to convey proximity: an incremental arrangement method. *ACM T Comput-Hum Int* 1997; 4: 197–229.
66. Koren Y, Carmel L and Harel D. ACE: a fast multiscale eigenvectors computation for drawing huge graphs. *Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02)*, Boston, Massachussets, 28-29 October 2002; 137–144, IEEE.
67. Frishman Y and Tal A. Multi-level graph layout on the GPU. *IEEE T Vis Comput Gr* 2007; 13: 1310–1319.
68. Gajer P and Kobourov SG. GRIP: graph drawing with intelligent placement. *J Graph Algorithm Appl* 2001; 6: 222–228.
69. Hadany R and Harel D. A multi-scale algorithm for drawing graphs nicely. *Discrete Appl Math* 2001; 113: 3–21.
70. Godiyal A, Hoberock J, Garland M, et al. Rapid multipole graph drawing on the GPU. In: Tollis I and Patrignani M (eds) *Graph drawing, lecture notes in computer science*, vol. 5417. Berlin/Heidelberg: Springer, 2009, pp. 90–101.
71. Koren Y, Carmel L and Harel D. Drawing huge graphs by algebraic multigrid optimization. *Multiscale Model Sim* 2003; 1: 645–673.
72. Hall KM. An r-dimensional quadratic placement algorithm. *Manage Sci* 1970; 17: 219–229.
73. Davidson GS, Hendrickson B, Johnson DK, et al. Knowledge mining with VxInsight: discovery through interaction. *J Intell Inf Syst* 1998; 11: 259–285.
74. Chen L and Buja A. Energy/stress functions for dimension reduction and graph drawing: power laws and their clustering properties. 2009, http://stat.yale.edu/~lc436/papers/Chen_Buja_energy_2009.pdf
75. Hachul S and Jünger M. An experimental comparison of fast algorithms for drawing general large graphs. In: Healy P and Nikolov N (eds) *Graph drawing, lecture notes in computer science*, vol. 3843. Berlin/Heidelberg: Springer, 2006, pp. 235–250.
76. Aris A and Shneiderman B. Designing semantic substrates for visual network exploration. *Inform Visual* 2007; 6: 281–300.
77. Misue K, Eades P, Lai W, et al. Layout adjustment and the mental map. *J Visual Lang Comput* 1995; 6: 183–210.
78. Brandenburg FJ, Himsholt M and Rohrer C. An experimental comparison of force-directed and randomized graph drawing algorithms. In: *Proceedings of the symposium on graph drawing, GD '95*, Passau, Germany, 20-22 September 1995, pp. 76–87, London, UK: Springer-Verlag.
79. Van Ham F and Rogowitz B. Perceptual organization in user-generated graph layouts. *IEEE T Vis Comput Gr* 2008; 14: 1333–1339.
80. Lee B, Plaisant C, Parr CS, et al. Task taxonomy for graph visualization categories. In: *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization, BELIV '06*, Venice, Italy, 23 May 2006, pp. 1–5, ACM.
81. Gansner ER and North SC. An open graph visualization system and its applications to software engineering. *Software Pract Exper* 2000; 30: 1203–1233.
82. Brandes U and Wagner D. Using graph layout to visualize train interconnection data. In: Whitesides S (ed.) *Graph drawing, lecture notes in computer science*, vol. 1547. Berlin/Heidelberg: Springer, 1998, pp. 44–56.
83. Finkel B and Tamassia R. Curvilinear graph drawing using the force-directed method. In: Pach J (ed.) *Graph drawing, lecture notes in computer science*, vol. 3383. Berlin/Heidelberg: Springer, 2005, pp. 448–453.

84. Gehlenborg N, O'Donoghue SI, Baliga NS, et al. Visualization of omics data for systems biology. *Nat Methods* 2010; 7: s56–s68.
85. Yildirim MA, Goh KI, Cusick ME, et al. Drug-target network. *Nat Biotechnol* 2007; 25: 1119–1126.
86. Heer J and Perer A. Orion: a system for modeling, transformation and visualization of multidimensional heterogeneous networks. In: *Visual analytics science and technology (VAST)*, 2011.
87. Freeman LC. Visualizing social networks. *J Soc Struct* 2000; 1.
88. Moreno JL. *Who will survive*. Nervous and Mental Disease Publishing Company, 1934; Washington D.C., USA.
89. Lundberg GA and Steele M. Social attraction-patterns in a village. *Sociometry* 1938; 1: 375–419.
90. Northway ML. A method for depicting social relationships obtained by sociometric testing. *Sociometry* 1940; 3: 144–150.
91. Brandes U, Kenis P and Wagner D. Communicating centrality in policy network drawings. *IEEE T Vis Comput Gr* 2003; 9: 241–253.
92. Sugiyama K, Tagawa S and Toda M. Methods for visual understanding of hierarchical system structures. *IEEE T Syst Man Cyb* 1981; 11: 109–125.
93. Brandes U, Raab J and Wagner D. Exploratory network visualization: simultaneous display of actor status and connections. *J Soc Struct* 2001; 4.
94. Koren Y and Harel D. One-dimensional layout optimization, with applications to graph drawing by axis separation. *Comput Geom* 2005; 32: 115–138.
95. Krzywinski M, Birol I, Jones SJ, et al. Hive plots – rational approach to visualizing networks. *Brief Bioinform* 2011. DOI 10.1093/bib/bbr069.
96. He W and Marriott K. Constrained graph layout. *Constraints* 1998; 309: 1895–1908.
97. Dwyer T and Koren Y. DiG-CoLa: directed graph layout through constrained energy minimization. In: *IEEE symposium on information visualization*, Minneapolis, Minnesota, USA, 23-25 October 2005, pp. 65–72, IEEE.
98. Dwyer T, Koren Y and Marriott K. Drawing directed graphs using quadratic programming. *IEEE T Vis Comput Gr* 2006; 12: 536–548.
99. Dwyer T, Koren Y and Marriott K. Stress majorization with orthogonal ordering constraints. In: Healy P and Nikolov N (eds) *Graph drawing*, *lecture notes in computer science*, vol. 3843. Berlin/Heidelberg: Springer, 2006, pp. 141–152.
100. Dwyer T, Koren Y and Marriott K. IPSep-CoLa: an incremental procedure for separation constraint layout of graphs. *IEEE Vis Comput Gr* 2006; 12: 821–828.
101. Barsky A, Munzner T, Gardy J, et al. Cerebral: visualizing multiple experimental conditions on a graph with biological context. *IEEE T Vis Comput Gr* 2008; 14: 1253–1260.
102. Jianu R, Yu K, Cao L, et al. Visual integration of quantitative proteomic data, pathways, and protein interactions. *IEEE T Vis Comput Gr* 2010; 16: 609–620.
103. Dwyer T, Marriott K, Schreiber F, et al. Exploration of networks using overview + detail with constraint-based cooperative layout. *IEEE T Vis Comput Gr* 2008; 14: 1293–1300.
104. Dwyer T, Marriott K and Wybrow M. Dunnart: a constraint-based network diagram authoring tool. In: Tollis IG and Patrignani M (eds) *Graph drawing, lecture notes in computer science*, vol. 5417. Berlin/Heidelberg: Springer, 2009, pp. 420–431.
105. Dwyer T, Marriott K and Wybrow M. Topology preserving constrained graph layout. In: Tollis I and Patrignani M (eds) *Graph drawing, lecture notes in computer science*, vol. 5417. Berlin/Heidelberg: Springer, 2009, pp. 230–241.
106. Schreiber F, Dwyer T, Marriott K, et al. A generic algorithm for layout of biological networks. *BMC Bioinformatics* 2009; 10: 375.
107. Jourdan F and Melancon G. A tool for metabolic and regulatory pathways visual analysis. *Visualization and Data Analysis* 2003; 46–55.
108. Spritzer A and Freitas CMDS. Design and evaluation of Magnetviz – a graph visualization tool. *IEEE T Vis Comput Gr* 2011; 18: 822–835.
109. Shneiderman B and Aris A. Network visualization by semantic substrates. *IEEE T Vis Comput Gr* 2006; 12: 733–740.
110. Masui T. Graphic object layout with interactive genetic algorithms. In: *Proceedings IEEE workshop on visual languages*, Seattle, Washington, USA, 15-18 September 1992, pp. 74–80, IEEE .
111. Kosak C, Marks J and Shieber S. Automating the layout of network diagrams with specified visual organization. *IEEE T Syst Man Cyb* 1994; 24: 440–454.
112. Branke J, Bucher F and Schmeck H. A genetic algorithm for drawing undirected graphs. In: *Proceedings of the third Nordic workshop on genetic algorithms and their applications*, Helsinki, Finland, 18-22 August 1997, pp. 193–206, Department of Information Technology and Production Economics, University of Vaasa.
113. Rodrigues EM, Milic-Frayling N, Smith M, et al. Group-in-a-box layout for multi-faceted analysis of communities. In: *Third IEEE conference on social computing*, MIT, Boston, USA, 9-11 October 2011.
114. Fekete JD, Wang D, Dang N, et al. Overlaying graph links on treemaps. In: *IEEE symposium on information visualization 2003 poster compendium*, Seattle, Washington, USA, 19-21 October 2003, pp. 82–83, IEEE.
115. Muelder C and Ma KL. A treemap based method for rapid layout of large graphs. In: *2008 IEEE pacific visualization symposium*, Kyoto, Japan, 5-7 March 2008, pp. 231–238, IEEE.
116. Muelder C and Ma KL. Rapid graph layout using space filling curves. *IEEE T Vis Comput Gr* 2008; 14: 1301–1308.
117. Itoh T, Muelder C, Ma KL, et al. A hybrid space-filling and force-directed layout method for visualizing multiple-category graphs. In: *2009 IEEE pacific visualization symposium*, Beijing, China, 20-23 April 2009, pp. 121–128, IEEE.

118. Xu K, Cunningham A, Hong S-H, et al. Graphscape: integrated multivariate network visualization. In: *6th international Asia-Pacific symposium on visualization*, Sydney, Australia, 5-7 February 2007, pp. 33–40, IEEE .
119. Huang M and Eades P. *A fully animated interactive system for clustering and navigating huge graphs, lecture notes in computer science*, vol. 1547. Berlin/Heidelberg: Springer, 1998, pp. 374–383.
120. Quigley A and Eades P. Fade: graph drawing, clustering, and visual abstraction. In: Marks J (ed.) *Graph drawing, lecture notes in computer science*, vol. 1984. Berlin/Heidelberg:Springer, 2001, pp. 77–80.
121. Pretorius AJ and Van Wijk JJ. Visual analysis of multivariate state transition graphs. *IEEE T Vis Comput Gr* 2006; 12: 685–692.
122. Pretorius AJ. Visual inspection of multivariate graphs. *Comput Graph Forum* 2008; 27: 967–974.
123. Becker RA, Eick SG and Wilks AR. Visualizing network data. *IEEE T Vis Comput Gr* 1995; 1: 16–28.
124. Leydesdorff L and Persson O. Mapping the geography of science: distribution patterns and networks of relations among cities and institutes. *J Am Soc Inf Sci Tec* 2010; 61: 1622–1634.
125. Dharmawirya M. Germany pass-networks (against Australia) – 2010 FIFA World Cup 2010, http://scientometrics.wordpress.com/2010/07/04/ger-pass-net work-vs-aus/ (accessed 6 October 2010).
126. Ericson M. When maps shouldn't be maps 2011, http://www.ericson.net/content/2011/10/when-maps-shouldnt-be-maps/ (accessed 16 December 2011).
127. Wattenberg M. Visual exploration of multivariate graphs. In: *Proceedings of the SIGCHI conference on human factors in computing systems,* CHI '06, Quebec, Canada, 22-27 April 2006, pp. 811–819, ACM Press.
128. Viau C, McGuffin MJ, Chiricota Y, et al. The FlowVizMenu and parallel scatterplot matrix: hybrid multidimensional visualizations for network exploration. *IEEE T Vis Comput Gr* 2010; 16: 1100–1108.
129. Elmqvist N, Dragicevic P and Fekete JD. Rolling the dice: multidimensional visual exploration using scatterplot matrix navigation. *IEEE T Vis Comput Gr* 2008; 14: 1141–1148.
130. Pretorius AJ and Van Wijk J. Multidimensional visualization of transition systems. *Proceedings of the Ninth International Conference on Information Visualization (IV05)* , London, UK, 6–8 July 2005; 323–328, IEEE.
131. Wu Y and Takatsuka M. Visualizing multivariate networks: a hybrid approach. In: *2008 IEEE pacific visualization symposium*, Kyoto, Japan, 5-7 March 2008, pp. 223–230, IEEE.
132. Wu Y and Takatsuka M. Visualizing multivariate network on the surface of a sphere. Misue K, Sugiyama K and Tanaka J (eds) *Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation* pp. 77–83, 2006; 60.
133. Dork M, Carpendale S and Williamson C. Visualizing explicit and implicit relations of complex information spaces. *Inform Visual* 2011; 1: 5–21 .
134. Gibson H and Faith J. Node-attribute graph layout for small-world networks. In: *15th international conference on information visualisation*, London, UK, 13–15 July 2011, pp. 482–487, IEEE .
135. Huang W and Eades P. *How people read graphs*. Darlinghurst, NSW, Australia: Australian Computer Society, Inc, 2005, pp. 51–58.
136. Salvini MM, Gnos A and Fabrikant SI. Cognitively plausible visualisation of network data. In: *Proceedings of the 25th international cartographic conference*, Paris, France, 3–8 July 2011.
137. Dwyer T, Lee B, Fisher D, et al. A comparison of user-generated and automatic graph layouts. *IEEE T Vis Comput Gr* 2009; 15: 961–968.
138. Purchase HC, McGill M, Colpoys L, et al. Graph drawing aesthetics and the comprehension of uml class diagrams: an empirical study. In: *Proceedings of the 2001 Asia-Pacific symposium on information visualisation*, Sydney, Australia, 3–4 December 2001, pp. 129–137, Australian Computer Society .
139. Purchase HC, Carrington D and Allder JA. Empirical evaluation of aesthetics-based graph layout. *Empir Softw Eng* 2002; 7: 233–255.
140. Becker MY and Rojas I. A graph layout algorithm for drawing metabolic pathways. *Bioinformatics* 2001; 17: 461–467.
141. Von Landesberger T, Kuijper A, Schreck T, et al. Visual analysis of large graphs: State-of-the-Art and Future Research Challenges, *Computer Graphics Forum*, 2011; 30(6): 1719–1749, Blackwell Publishing Ltd.