James Cowling **jcowling@cs.usyd.edu.au**

The University of Sydney

School of Information Technologies

COMP 3809 Software Project (Advanced)

WilmaScope Fast Layout Engine Product Development Report

Abstract: The development process for the implementation of a fast graph layout engine is examined. This implementation is of an algorithm proposed by Davidson, Wylie and Boyack in their paper "Cluster Stability and the Use of Noise in Interpretation of Clustering"[7]. A critical analysis is given of the implementation produced, along with the development process adopted. The implementation has provided a valuable insight into the effectiveness of the said algorithm in practice. It has been discovered from the implementation that the algorithm performs less well than hoped, a fact that has been noted and further investigated.

Introduction

This report gives an account of the development of an implementation of a fast graph placement algorithm for the purposes of the Information Visualisation Research Group in the School of Information Technologies at the University of Sydney.

The implementation produced, as well as the process undertaken in producing this implementation, will be discussed in detail. A thorough evaluation of these aspects will be entered into, with a description of the knowledge and experience gained through the project.

The context of the project will be given first, to establish a grounding on which to base the remainder of the report. Following the context will be a discussion of the algorithm, which spans both the context of the project and the project implementation itself. A thorough understanding of the algorithm was required to produce a product that not only implemented the algorithm as presented, but also facilitated for research into further development in the area.

Project Context

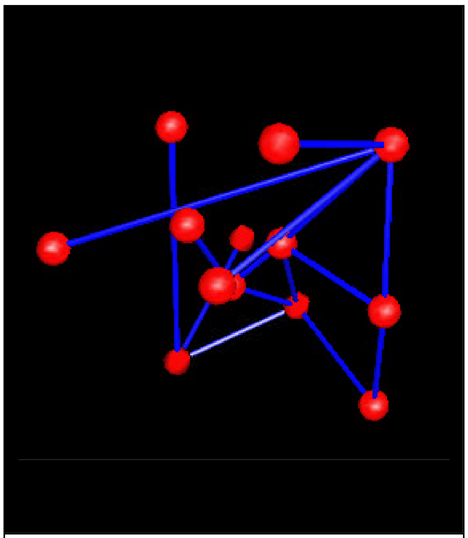
Background on the Information Visualisation Research Group

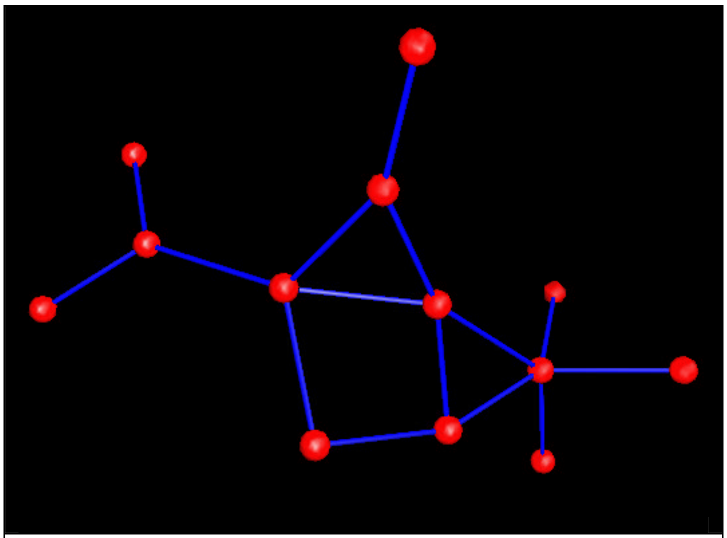
The Information Visualisation Research Group is a research group in the School of Information Technology at the University of Sydney. The group was formed in January 2000 and is lead by Professor Peter Eades, who is also the head of the School. The objective, or mission, of the group is to make "good pictures of abstract information, such as stock prices, family trees, and software design diagrams"[1]. These pictures can then be used to convey information in an effective way, and to provide a rapid means for evaluating the information presented in the picture. The group see it as their "challenge"[1] to create algorithms that produce such pictures. Within the group is it seen that graphics hardware development over the past 15 years has offered a promise that the field of Information Visualisation can "deliver high quality pictures at a reasonable cost"[2]. The research done by the group investigates into "how to deliver this promise"[2].

Graph Layout Algorithms

A graph is an abstract structure that can be used to model relationships between a set of entities[3]. Graphs typically consist of a set of nodes or points, representing the entities, and links between these nodes representing the relationships between them. Various properties such as weight, natural length etc, may be associated with the links in the graph.

Such graphs can be extremely useful in visualising the information contained within them. The pictorial form is one that is very natural to the human observer and does not require any special knowledge or training to interpret. The usefulness of the graph however, is very much determined by its readability. The graph must be able to convey the meaning of the diagram rapidly and accurately.


A simple graph with a poor layout, obscuring the information presented.


The same graph conveying more information once presented with a good layout.

Traditionally, small graphs have been drawn by hand, with the creator of the graph responsible for ensuring it is represented in such a form that it will present the information clearly to the target audience[4]. As the size and complexity of graphs increases, the task of representing the graph in a clear manner becomes much more difficult. With the advent of computing technologies, software tools have been developed to help represent graphs in a manner that best displays the symmetries and structures in the underlying relationships.

The tools to manipulate a graph, so as to represent it in the best possible form, implement what are known as graph layout algorithms. They perform the task of positioning the nodes in a space of certain dimension, to represent the data in the clearest possible way. The class of algorithms most commonly used for generating the layout of a graph within the space are known as force-directed algorithms.

The force-directed class of graph layout algorithms are used frequently because they are able to effectively produce layouts considered 'nice' for small graphs. Note that the concept of 'niceness' in human terms is a combination of the aesthetics, symmetry and sense of structure of a graph, and is more of a psychological issue than one easily defined in concrete computing terms. To be able to organise the nodes in a 'nice' way, an algorithm must adopt some kind of metric, used to determine if a certain layout is better than another. This metric can only hope to evaluate a graph similarly to the complex factors used by a human and immediately obvious when looking at a graph.

The force-directed class of algorithms all share the common metric of an ‘energy’ property of the graph, and seek to minimise this energy[5]. The energy at a node is a direct result of the forces acted upon it by neighbouring nodes. Nodes connected to a particular node by links will exert an attractive force on the node, while neighbours not sharing a relationship with the node will exert a repulsive force, as would similarly charged particles in physical space. This energy metric is a rather broad concept and allows individual algorithms to take varied approaches in trying to produce effective graph layouts.

The major disadvantage of force-directed algorithms in general is than although they can be effective for smaller graphs, they are unable to scale to larger graphs, at least within practical constraints. This is because they usually rely on computing the force between each pair of nodes directly, a task which is very computationally expensive. Despite many efforts to implement complex techniques to improve scalability constraints, the class of force-directed algorithms have traditionally possessed quadratic scalability, and proved useless for graphs of significant size. This problem is addressed by a new breed for fast force-directed placement algorithms. The algorithm used in the project is a relatively new development and is described in the ‘Fast Layout Algorithm’ section of the report.

Impetus for Product Development

The product was developed for Tim Dwyer, a PhD student in the University of Sydney Information Visualisation Research Group. Tim’s research interests include “3D interactive graph visualisation systems and their application to financial data visualisation”[12]. Part of this research is funded by Capital Markets CRC[13] who wish to obtain the results of a study on financial market visualisation.

In recent times the size of data needed to be visualised has increased at a rate much faster than the visualisation technologies available[15]. A new requirement for increased speed of visualisation has also arisen due to the influence of streaming technologies and evolving data. The data held by Capital Markets CRC holds both these properties and the work done by Tim and researchers in the field aims at creating systems to visualise the data given these requirements. The product aims to assist in the development of such systems by providing a means for analysis of a potential solution to the problem.

To represent the visualisation of various data sources Tim has produced a graph visualisation system known as WilmaScope. This system has great capacity for interactive and aesthetic display of graph layouts in three dimensions. The default layout engine used by the visualisation package has high computational complexity attributes which forbid the implementation from representing the large graphs required to represent various real-world models. Tim is interested in investigating other alternatives to the currently used layout engine possessed by WilmaScope. There is a wide range of possible algorithms to be used for such a purpose, each of which must be evaluated in practice before implementing them for use in generating real-world graph layouts.

Constraints on resources forbid the testing of these algorithms in practice by current research staff. The project undertaken by myself aimed at enabling the research into a possible avenue for approaching graph layout within and without of the WilmaScope package. To satisfy these means an implementation of a particular new algorithm for generating layouts for very large graphs was required. This needed to provide the flexibility required in evaluating the algorithm over a range of parameters.

The project developed allows Tim to make a judgement on the effectiveness of the particular algorithm without having to devote the time and effort required to develop a working implementation himself. Without such a project the investigation of such an option would not have been feasible. The extensibility provided by the implementation also allows further research into the algorithm beyond that initially intended, with a thorough study of the response of the algorithm to varied parameters now possible. It also facilitates the relative ease of development of another alternative layout algorithm, which could be then compared with the current implementation. This would produce very valuable research results.

Addressing of Goals

As a result of the project, research into the feasibility of the use of such an algorithm has been possible that otherwise would not have been. As such, the project has not only assisted in current research, but opened up new research direction which would have otherwise been infeasible. The project has directly assisted in Tim Dwyer's goal of producing a graph visualisation system to handle graphs of significant size. It has allowed him an insight into the feasibility of the algorithm implemented beyond that originally available.

Also of significant importance is the extension the project provides to the WilmaScope visualisation package. It is the first third-party module to be created for use by the system, and opens the door to further module development by other parties. It provides an ideal vehicle for showcasing the modularity and extensibility of the package, while offering a useful addition to the package as a whole. The impact of this is such that the project will be included for download from the WilmaScope web page[14], and this report made available for public viewing.

As an effective mechanism for creating graph layouts, the project also directly addresses the goals of the research group in providing such layouts. It has also helped to shape further work into the visualisation provided by the group by guiding the choice of algorithm used in creating these visualisations.

A particularly interesting result of the project is the indication it gives as to the feasibility of the algorithm implemented. It was discovered through the product development that a number of weaknesses were present in the algorithm, weaknesses not acknowledged by the original research paper on the subject. The net result of these weaknesses has effectively ruled out the use of the particular algorithm in a practical context within the group. Although, insofar as immediate results are concerned, a product that demonstrated very good results from the algorithm would have been pleasing, the result obtained here is perhaps quite more important. A positive result from the implementation would have not lead to an absolute decision on the use of the algorithm, however the currently imperfect result points to a more definitive conclusion as to the future use of the algorithm. A thorough examination of these weaknesses in the algorithm is given in the algorithmic evaluation in this report.

Other Approaches to a Common Problem

A number of feasible alternatives exist to the algorithm implemented in the project. In a research context these algorithms would also be examined in detail, to address the relative merits of the algorithm as implemented. A brief overview of some of the more promising techniques for fast layout will be given to demonstrate a working knowledge of the approaches used in attacking the problem of creating a layout for very large graphs.

Walshaw[5] discusses a method for generating a graph layout by using multiple levels of clustering. In his approach, groups of nodes are merged together to clusters, with this process repeating on the clusters until the size of the graph falls below a specified threshold. This super-graph is then given a layout using traditional force-directed techniques. Once this layout is complete, the clusters are expanded one level and the process again repeats until a layout for the original graph results. It is claimed to both accelerate and give a "more global quality to the force-directed placement"[5]. The algorithm appears to function very well in practice and perform up to an order of magnitude faster than other current techniques.

Harel and Koren[3] propose a very similar method where instead of combining entire groups of nodes, nodes drawn close to each other aand joined by a common edge are merged together, the process repeating until a sufficiently simple graph is formed. This super-graph is laid out much like the Walshaw technique although an additional algorithm is used for local beautification of the layout – that proposed by Kamada and Kawai[11]. This has the additional appeal of the enhancement of local details in the layout, however given the results it appears the Walshaw algorithm may function equally as well in practice without the added complexity.

Gajer, Goodrich et al[4] discuss an interesting approach to solving the problem of artefacts in the resulting layout. They propose a technique where a layout is generated for a graph in a given dimension, higher than the dimension intended for display. They implement a variety of additional techniques, beyond the scope of this initial analysis, to create this higher dimensional layout within realistically small time bounds. This layout is then projected onto two or three dimensions, creating a resulting layout that contains a much higher degree of global symmetry and 'smoothness'. This is an approach that could be used in addition to a great variety of other layout methods, to enhance the quality of the resulting layout. A somewhat similar technique is available in the current project implementation, albeit with the restriction of projecting three dimensions onto two.

Fast Layout Algorithm

The algorithm implemented in the development of the fast layout engine for WilmaScope is that discussed by Davidson, Wylie et al. in their paper "Cluster Stability and the Use of Noise in Interpretation of Clustering"[7]. This algorithm was based on earlier work by Fruchtermann and Rheingold[8]. It belongs to the class of force-directed placement algorithms however it differs from most of its alternatives in one main area, in that it possesses linear time scalability as opposed to the quadratic scalability plaguing the majority of force-directed algorithms. The algorithm will be described in general before a critical analysis of its implementation is given.

Davidson et al[7] develop their algorithm based on 4 main criteria, which can be seen as the goals for the algorithm. These criteria are used to develop a tight set of algorithm requirements, each of which is addressed by a separate feature of the algorithm. They are as follows:

1. Vertices connected by a common edge should be drawn near each other.
2. Non-connected vertices should be forced away from each other.
3. The results of the algorithm should be insensitive to random starting conditions.
4. The complexity of the computation should be at a minimum.

Although very simple in their specification, these criteria allow the algorithm designers to abstract a great detail of the complexity of graph drawing and break the algorithm down into 4 simple goals, each of which will be approached somewhat individually. The methods for satisfying each of these criteria will be discussed in relation to the fast force-directed placement engine for WilmaScope.

Calculating Energy of a Node (criteria 1 & 2)

As in all force-directed placement algorithms, the position of a node in the space is evaluated by calculating an energy term for that node and attempting to minimise the total energy of the system. Continuing the concept of energy, criteria 1 and 2 as described above are accommodated via a 'force' term for both repulsion and attraction. Nodes connected with each other exert an attractive force on one another, while disconnected nodes repel. The calculation of an attractive and repulsive force term was described by Fruchtermann et al[8], however in this approach, the two forces are combined into a single scalar energy quantity.

Most, and indeed the more computationally expensive, force-directed placement algorithms utilise the attractive and repulsive forces from neighbouring nodes to calculate a force vector for the current node, which is then used to direct the node towards a position of lower energy. In the interests of computational efficiency however, Davidson et al[7] propose simply calculating an energy value for the node. The nodes are moved around the space randomly, maintaining their position if they achieve a decrease in energy. This simplistic manner of positional evaluation is enhanced by an advanced process of simulated annealing, to be described further, which places constraints on the random movements.

The suggested equation for calculating the energy at a node then is given by the following equation:

$$K_{i(x,y)} = \left[\sum_{j=1}^{n_i} \left(w_{i,j} \times l_{i,j}^2 \right) \right] + D_{x,y}$$

$K_{i(x,y)}$	= The energy of node *i* at location *x, y*
n_i	= The degree (number of edges) at node *i*
$w_{i,j}$	= The weight of the edge between nodes *i* and *j*
$l_{i,j}^2$	= The squared distance between nodes *i* and *j*
$D_{x,y}$	= A density term representing the repulsion force at position *x, y*

It can be clearly seen that the first half of the above equation, that concerning the attractive force between nodes, has time requirements depending solely on the number of edges connected to a given node. This quantity should remain relatively constant within the same type of graph, independent of the number of nodes. Thus the scalability of the first term, over all the nodes in the graph, is linear in the number of nodes.

The second term however is less easy to obtain analytically within linear time bounds. It is an area in which a novel approach is taken to ensure constant time determination of this factor for each node. If time constraints were not a concern then this term could be calculated accurately simply using an equation such as the following, over all of the nodes.

$$D_i = \sum_{\substack{j=1 \\ i \neq j}}^{|N|} d_{i,j}$$

D_i	= Repulsion force at node *i*
$\lvert N \rvert$	= The number of nodes in the graph
$d_{i,j}$	= The (Euclidean) distance between nodes *i* and *j*

This equation yields a quadratic scalability over all the nodes in the graph, which is unacceptable for our purposes. Since the goals for computational efficiency so heavily affect the choice of calculation of this density term, its calculation will be discussed when addressing criteria 4.

As mentioned previously, the nodes are moved around the space somewhat randomly seeking positions of lower energy. From a simplified viewpoint, the basic concept can be seen as:

1. Moving a node to a new, random, position
2. Evaluating the energy at the new position
3. If the previous energy was lower, returning the node to its original position.

This very basic concept does not look to give good results, and in fact it does not, at least in comparison with more complex force-directed techniques. The concept does have the advantage of being very fast however, and this attribute is exploited while incorporating a variety of constraints on the movement which are able to yield much better results in practice.

A technique similar to that of 'simulated annealing'[9] is used to ensure that the energy of the system is minimised in an ordered way. This also aims to produce the greatest possible reduction of energy. The energy function for each node is minimised gradually, over three distinct phases. The three phases and their properties are summarised as follows:

- Boiling Phase:
    - In this phase the nodes are allowed to expand out into the general area which they will eventually occupy.
- Quenching Phase:
    - The maximum allowable distance jumped by each node is gradually decreased to minimise the energy equations.
- Simmering Phase:
    - The nodes are restricted to jumping small distances to make detailed local changes.

The phases are made more complex by the addition of the barrier jumping technique which will be described when addressing attribute 3.

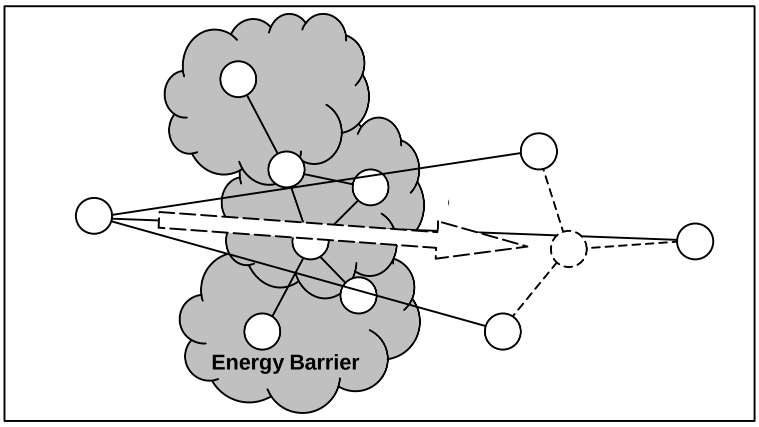
While much more complicated methods were considered for calculating the new position of each node, as an alternative to simple random jumps, these have been found to yield poor results in practice due to their complexity and the chaotic nature of graph organisation. Such methods as gradient descent, and those with momentum terms, were raised by Davidson et al[7] but found to be less effective in practice.

Ensuring stability with respect to initialization (criteria 3)

A problem plaguing similarly 'random' layout algorithms is that the graph can very rapidly settle into local minima of which it will not recover from if reduction of energy is the only criteria for movement of the nodes. This problem is particularly likely during the early stages of the algorithm where the node positions are expanding to fill the locations they will finally reside in. With dense, and in particular, very large graphs that are being operated on, a node can quite easily become 'stuck' behind a large group of other nodes, their combined density forming a barrier through which the node cannot pass. The problem of nodes settling into local minima is quite easily seen as 'folding' of a graph, where sections of a graph become hopelessly tangled. Analysis of this folding will be given in the evaluation of the project.

This tendency to local minima results in a layout that is very dependent on the initial positions of the nodes, and will most likely produce a result that is both aesthetically unpleasing and obscures the underlying structure of the graph. The extreme sensitivity of the layout to initial conditions contradicts our third criteria, and thus steps must be taken to avoid it.

To avoid becoming trapped in a local minimum, a node must occasionally move to a position of higher energy, if that new state will lead to a layout that ultimately leads to decreased energy overall. A technique known a 'barrier jumping'[7] is used to avoid this local minima problem, where instead of moving to minimise its own energy, the algorithm will ignore the repulsion term for that node and move it directly to a position which minimises the attractive force. This jump is performed by simply moving the node directly to its barycentre, defined as the centre of all connected neighbours, taking into account edge weights. A diagram of this process is given as follows:



The implementation of barrier jumping has a side effect in that it causes the graph to 'collapse' upon itself, with the nodes moving in towards each other each time they move to the centre of their neighbours. This phenomenon must be avoided since it will destroy the balanced layout of the graph and defeat the purpose of the implementation of the technique to begin with. Thus only a certain number of nodes undergo barrier jumping, and this number corresponds to the phase that the algorithm is in. Barrier jumping is most important during the boiling phase where the correcting nature of the technique is essential to prevent poor initial placement stemming from a random starting condition. The requirement for barrier jumping decreases as the algorithm progresses, and the graph reaches a more stable state. For this reason the rate of barrier jumping must decrease during the simmering phase. Davidson et al[7] recommend a rate of 25% during the boiling phase, declining linearly during the quenching phase to 10%. Barrier jumping is not used at all during the simmering phase since at that stage the graph has already reached its general shape and simply needs detailed corrections to produce a pleasing layout.

Reducing computational complexity (criteria 4)

It has been shown that linear $\Theta(|N|)$ time can be easily achieved in calculating the attraction term in the energy equation. The brute force method for calculating the repulsion term however possesses quadratic $\Theta(|N|^2)$ time constraints. This is too expensive for real-world applications and instead an alternative method for calculating the repulsion force was created.

The method for calculating the repulsion term is a compromise between accuracy and efficiency. Instead of calculating a specific repulsion term at a node, the repulsion force is calculated as the general density in the area surrounding the node. Here, instead of the nodes being repelled by disconnected neighbours, they are repelled by only an overcrowding, simply a repulsion force due to all the nodes in its surroundings, regardless of whether they are connected neighbours or not. This method gives quite similar results to the more computationally complex brute force one, and appeals logically. The density at a point in space is then determined using a grid-based technique.

Fruchtermann[10], proposes a binning technique to determining a local neighbourhood of nodes with which to calculate an estimate of the repulsion force at a node. However, this approach requires a uniformly distributed graph in order to produce results approaching linear time over all nodes. As the algorithm as used assumes no such uniform distribution, and given a highly connected graph may encounter quite non-uniform distribution, a more general technique was used in favour of binning.

In the more general, density field[7], algorithm, each node places an energy 'footprint' on a grid or array. Once each node has placed its energy footprint on the grid, the energy density at a specific point can be looked up in constant time. When a node changes position, its old footprint must be subtracted from the grid, and the new one added. If this technique is followed, the overhead is simply one of constant time when changing a nodes position, giving a scalability of $\Theta(|N|)$ over all of the nodes. This helps to satisfy the fourth criteria of keeping computational complexity to a minimum, with linear scalability being the theoretical minimum to calculate a layout of the graph.

Metric for Success

Note that in calculating the new position of a node, only the energy at that particular node is considered, the effect on the entire system is not taken into consideration. This 'greedy' approach to repositioning, while possibly seeming imperfect, proves to be a good approach in many fields of computing – with 'greedy' strategies giving good real-world results in comparatively small execution times. A metric for the entire energy of the system can be easily calculated however, and is given as:

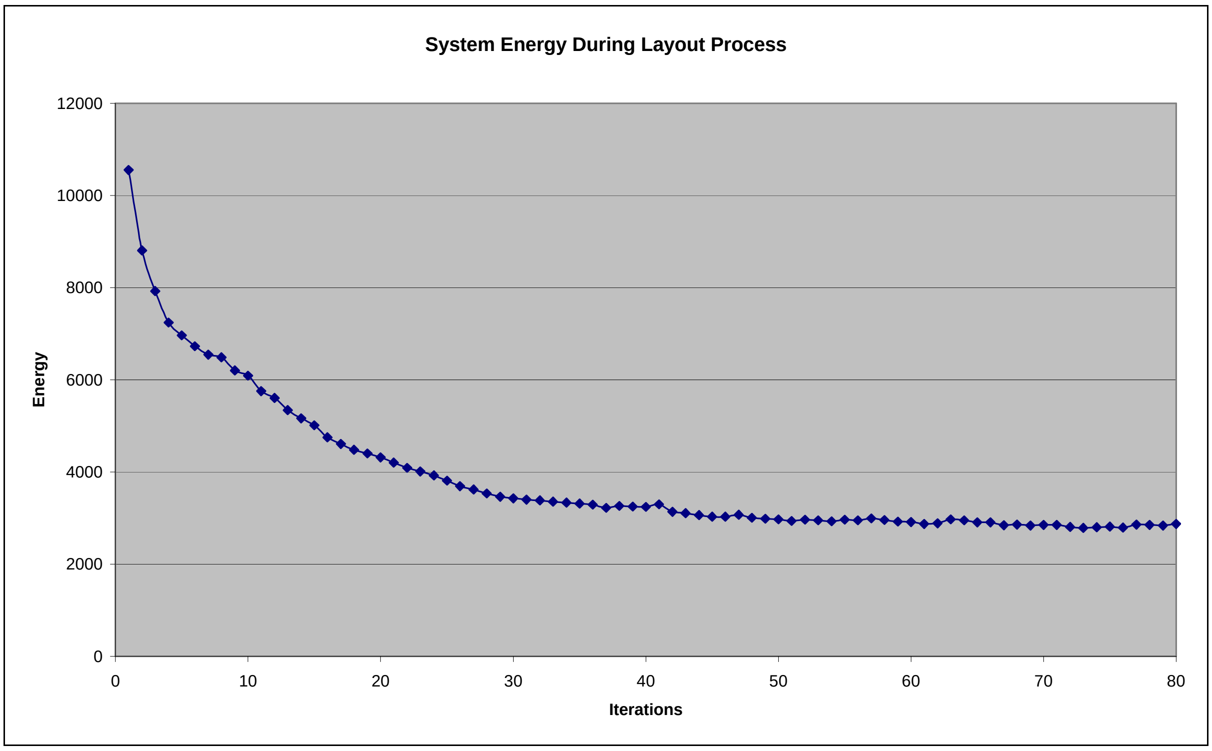
$$TotalEnergy(G) = \sum_{i=1}^{|N|} K_i$$

$TotalEnergy(G)$	= The total energy of a graph G
K_i	= The energy at node *i*
$\lvert N \rvert$	= The number of nodes in the graph

The algorithm for laying out the graph will continue indefinitely unless terminated by some stopping condition. It is entirely possible to use this total energy metric as stopping criteria for the algorithm. However, this approach is highly dependent on establishment of the scale factors involved in calculating the energy at a node, as well as the individual node and edge weights. Although undoubtedly effective in practical use, the implementation as developed was for the purposes of research into this type of layout algorithm, which carries with it a unique set of project requirements. One of these is to be able to extensively modify the parameters involved, both between runs and at run-time. This flexibility means that the acceptable energy for a system, being the energy when it is in a stable state, can vary significantly and thus a different approach must be used.

To calculate the stopping criteria in this implementation, an approach proposed by Eades[9] is used. He suggests simply running the algorithm for a preset number of iterations. Given the capacity for varying this limit in the implementation, this approach allows just as accurate a termination criterion, while requiring much less overhead.

A graph of the system energy after each iteration of a sample run of the implementation is given. This illustrates the tendency of the graph to approach a practical minimum in regards to the total energy of the nodes. It is much simpler to identify a minimum number of iterations to bring the graph into the 'plateau' region of the graph, where the graph has reached a stable state, than it is to set an energy cut-off in advance, since the latter is highly dependent on the underlying structure of the graph.



Naïve Repositioning

An extension was made to the suggested algorithm to try to compensate for the high overheads incurred. One of the very expensive aspects of the algorithm is that during each iteration a node must be first moved into its new position before any evaluation is made of the possible gains. The energy at the new point is then calculated and if the energy is in fact higher, the node must be moved back to its original position. The trial-and-error nature of the algorithm on its own can be justified by the simplicity and intended speed of its approach; however when the error cases begin to pose a significant penalty, the algorithm is very much slowed down. As there is essentially random probability that a particular move will be an error, it becomes difficult to justify the appeal of an implementation that must spend a very large proportion of its time reversing erroneous moves it has made to nodes. Note that the moving of a node, while executed in constant time and not affecting the scalability of the algorithm, is the most expensive step involved, in regards to processing required. Although scalability of an algorithm is an important metric, it must also possess a low scalability *coefficient*, lest it be too slow in practice for any real-world use. In an attempt to mitigate the randomness of the algorithm, by reducing the error penalty, a new technique was developed for this particular implementation and dubbed "naïve repositioning".

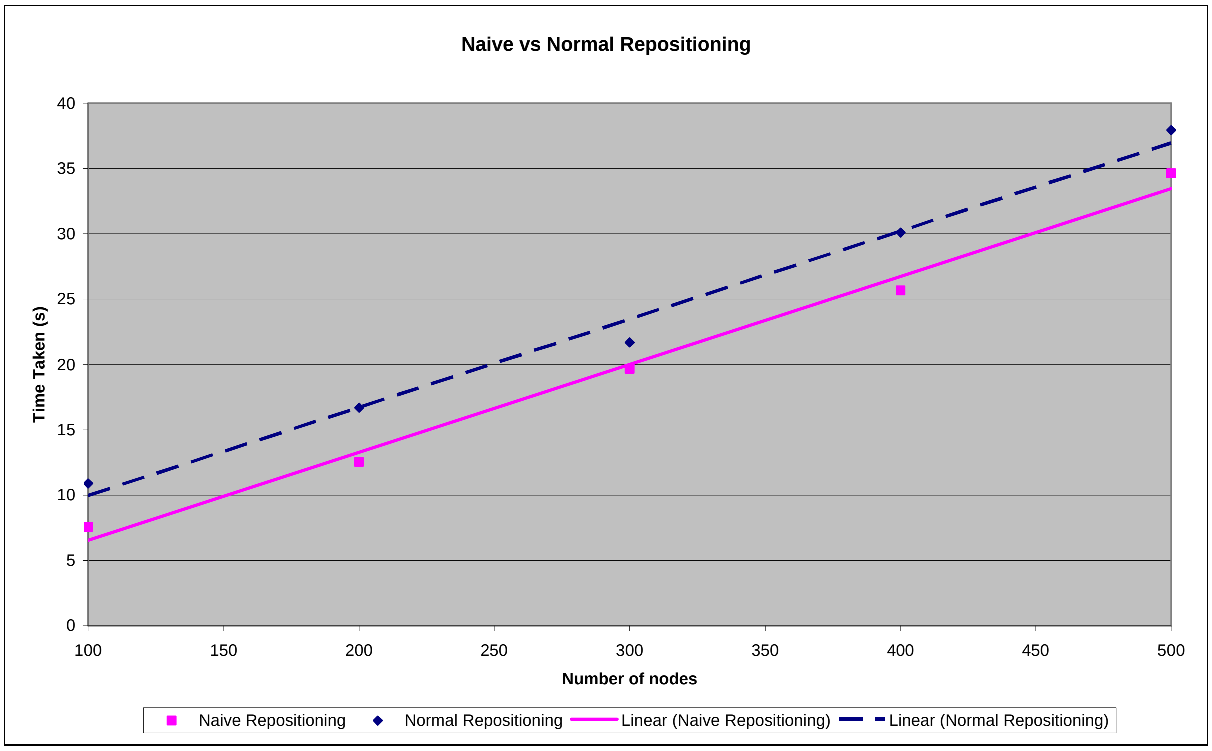
This technique is provided as an option to the standard method for evaluating a new position for each node. It appeals to the compromise made by the algorithm between accuracy and speed of execution. The density term concept, for example, adopts a simplified measure of the repulsion force in aid of reducing the computational complexity involved with more direct, exacting methods. Here this concept is taken a step further, where the algorithm makes a ‘guess’ as to whether a node will be better off in a certain position, before actually moving the node to that position, at least in terms of the density matrix. The normal and naïve repositioning methods are briefly summarised for comparison below.

Note that in the table, the “Move node to a new position” step is simply updating the coordinates stored by the node, and not actually updating the energy footprint of the node, the latter being the computationally expensive step.

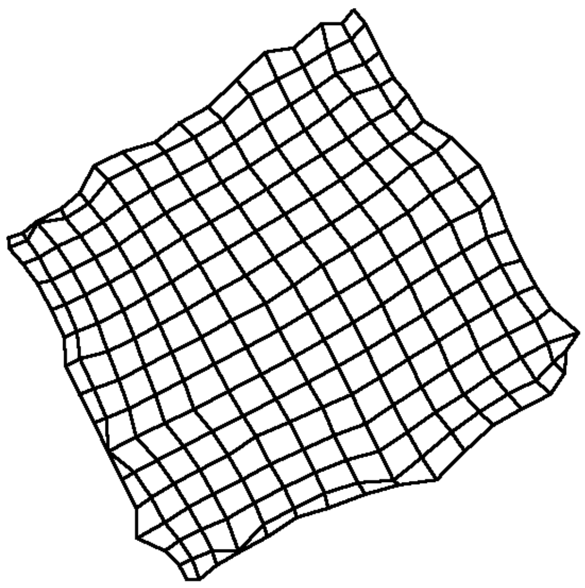
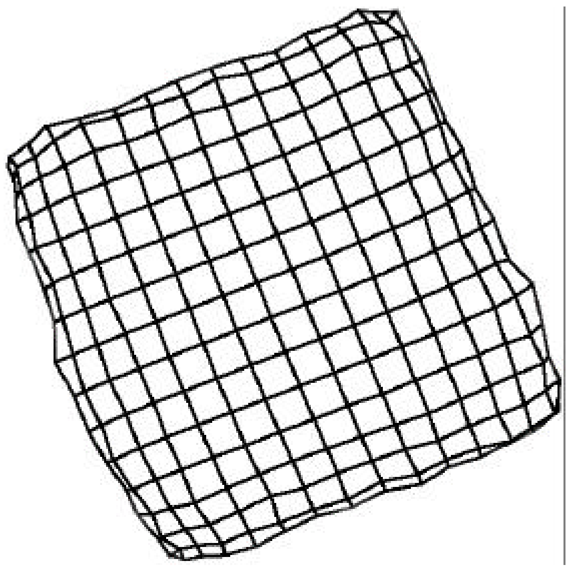
Normal Repositioning	Naïve Repositioning
Calculate current node potential	Calculate current node potential
Move node to new position	Move node to new position
Remove previous energy footprint and place new one	
Evaluate new potential	Evaluate new potential
	Add point density for mass of the node to the potential
if current potential is greater than original potential	
Move node back to original position	Move node back to original position
Remove new energy footprint and place old one again	

The step of adding the point density for the mass of the node to the potential calculated is an attempt to estimate what the density would be at the new position. This is, in general, a very accurate estimation. The point density is simply the density that the node will place in its centre, regardless of the attenuating density radiating outwardly from the node, which is a constant lookup.

The naïve repositioning method gives slightly less accurate results in practice since it does not adjust the density for the old position before evaluating the new position. This means that if the two positions are spatially close the results will be skewed by the effect of the energy footprint of the old position. The compromise however does allow the avoidance of removing and adding an energy footprint to the density matrix, a process that is quite expensive due to the many array accesses required. If the nodes are sufficiently far apart they will be moved as accurately in naïve repositioning as they would with the standard repositioning method. A brief analysis of the performance of these two methods has been made with the results as follows:



It can be seen from the plot that both naïve and normal repositioning possess identical scalability. However, it is readily observable that naïve repositioning functions faster in practice. While the time difference may seem insignificant, the gains can become quite large when the resolution and footprint size are set high in order to obtain more accurate results than the relatively fast settings used for testing above. As a justification of the accuracy of naïve repositioning, two sample layouts are given below, the first using the standard repositioning method and the second using naïve repositioning.

	
16x16 Grid – Normal Repositioning	**16x16 Grid – Naïve Repositioning**

The naïve repositioning function may be switched on and off during run-time. For this reason it is suggested that it is used during the early repositioning stages where the nodes are simply expanding outwards and making very general moves, while the normal reposition used for the latter phases of the layout where nodes may reside in close proximity to each other and fine detail is required.

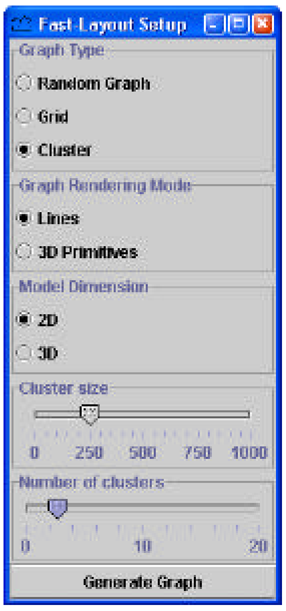
Product Implementation

The product was written in Java to be used from Tim Dwyer's WilmaScope graph visualisation system. As such the product is not a stand-alone application but rather a module which can 'plug into' the existing package to allow the evaluation of the algorithm in practice. The nature of the product as part of an existing, larger, package carried with it additional requirements not held by a product that may function in isolation.

The implementation adopts the structural conventions of the existing package in terms of code design, and is written to be as extensible and modular as possible. It consists of a package that abstracts away the details of the layout, and instead accommodates a standard interface for initialising the graph and requesting a layout with given parameters. A test engine was also integrated with the WilmaScope package as a whole to allow the automatic generation of test graphs when experimenting with the implementation in a research setting.

General Interface

Two frames are provided to the end user, one for controlling the initialising of the graph with the other for manipulating the layout engine parameters before and during run-time. These frames appear as follows, with a description following:


Fast-Layout Setup

Graph Type: Random Graph, Grid, [✓ Cluster]

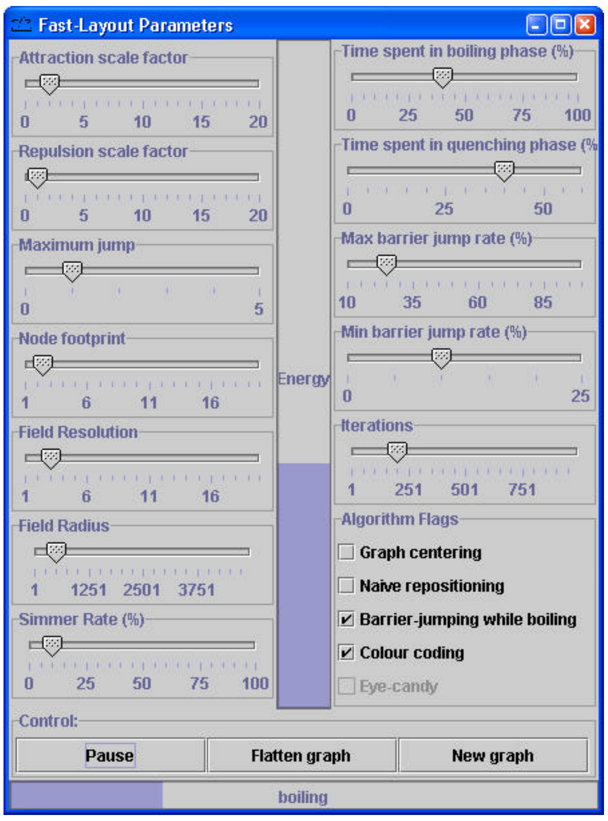
Graph Rendering Mode: [✓ Lines], 3D Primitives

Model Dimension: [✓ 2D], 3D

Cluster size (0 250 500 750 1000)

Number of clusters (0 10 20)

Graph Initialisation User Controls


Fast-Layout Parameters

Attraction scale factor (0 5 10 15 20)

Repulsion scale factor (0 5 10 15 20)

Maximum jump (0 5)

Node footprint (1 6 11 16)

Field Resolution (1 6 11 16)

Field Radius (1 1251 2501 3751)

Simmer Rate (%) (0 25 50 75 100)

Energy

Time spent in boiling phase (%) (0 25 50 75 100)

Time spent in quenching phase (% (0 25 50)

Max barrier jump rate (%) (10 35 60 85)

Min barrier jump rate (%) (0 25)

Iterations (1 251 501 751)

Algorithm Flags: Graph centering, Naive repositioning, [✓ Barrier-jumping while boiling], [✓ Colour coding], Eye-candy

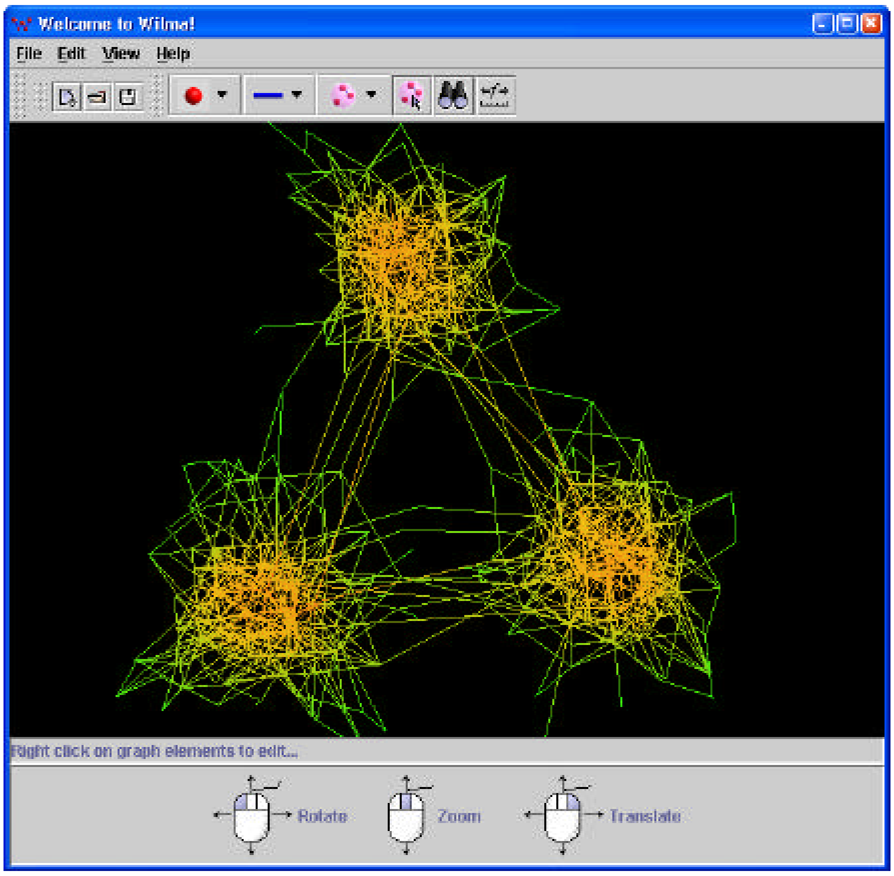
Control:

Engine Parameter User Controls

The graph initialisation controls allow the user to create a randomly generated graph of a certain type. The user can create purely random, grid, or clustered graphs. The grid option is particularly useful for analysing the response of the algorithm to folding of the graph, while the clustered approach is able to indicate the performance of the algorithm in resolving clustered graph sections. Within these three options the user is able to set the size of the graph produced. Options are also provided for line or 3D-primitive rendering modes. In the former, the graph is rendered simply with lines drawn representing edges. This is fast and good for analysing the general performance of the implementation in generating a layout for a large graph. When using the 3D-primitive rendering mode each node is represented by a three-dimensional sphere, with the edges represented by cylinders. This option, despite being much more aesthetically pleasing, conveys a greater sense of depth in the graph and is useful for analysing the layout of particular sections of the graph or identifying the energy at certain nodes.

The engine parameter controls allow the user to modify any parameter affecting the execution of the algorithm implementation, both before the layout process has begun and during run-time. When these parameters are modified, via convenient slider bars, the implementation will automatically scale its current state attributes to accommodate the new settings, and continue seamlessly with the graph layout. A set of check-boxes are provided to the user to enable or disable features such as naïve repositioning, graph centring, or colour-coding – features that will be elaborated on. Simple buttons are provided to run or pause the layout process, flatten the graph onto two dimensions, or create a new graph. In addition to the controls provided, the user is also presented with two status indicators – a progress bar representing the percentage of the layout completed and the current state of execution, and also a gauge showing the total energy of the system to give an indication of the practical progress made by the layout engine.

During the execution of the implementation the user is presented with an animated representation of the graph layout in process. They may rotate, zoom, or scale this visual representation in real time while the layout engine runs. Once this process is complete the user is notified and the graph layout is presented for their inspection. A sample representation of a graph layout within the WilmaScope package follows.



Right click on graph elements to edit...

The layout of a clustered graph, in line rendering mode, as seen by the user.

User Options

A number of options affecting the layout engine and the graph visualisation are available to the user, as mentioned previously, and will be given a brief description.

An option is provided to recentre the graph after each iteration. This allows a decision by the user to make a compromise between the inefficiency of moving the graph to the middle of the available ‘universe’, and the extra memory required to store the densities of a graph that strays a great distance from the initial centre. Another simple option provided is to enable or disable barrier jumping during the boiling phase, to allow for experimentation in the use of the algorithm.

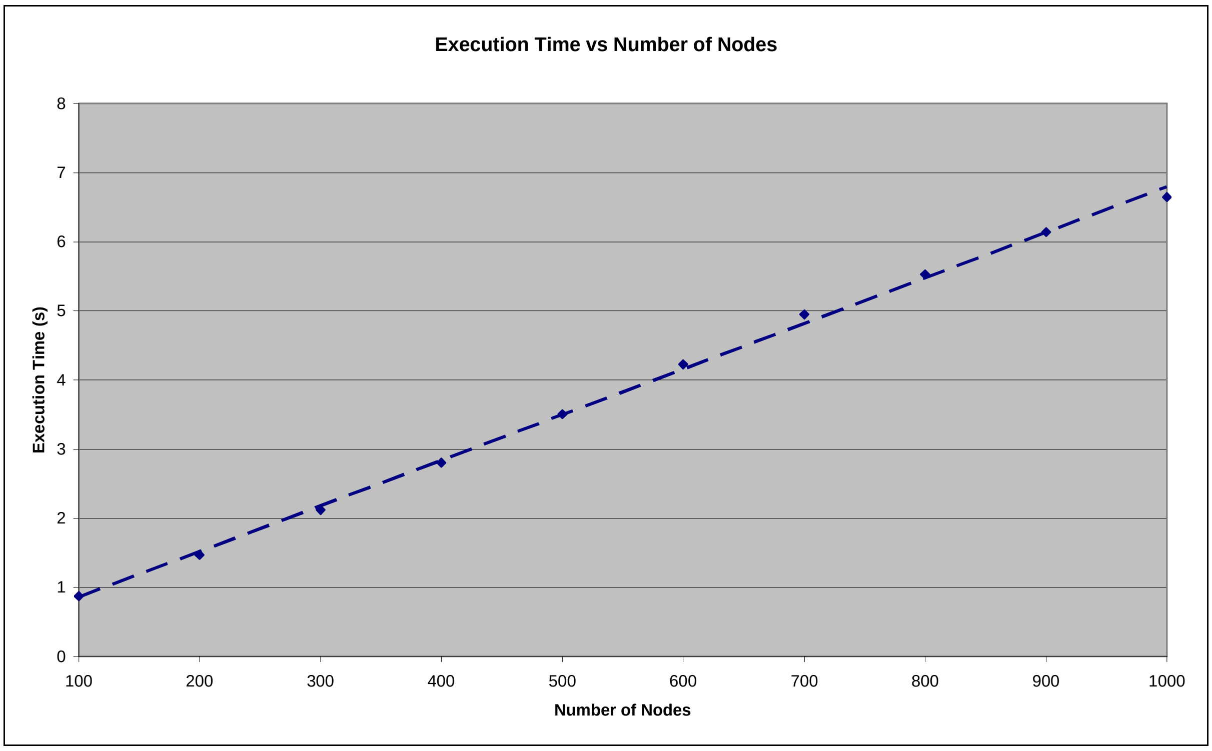
The option to flatten the graph onto two dimensions is relevant when investigating the effects of projecting a higher dimensional layout onto a lower dimension. This projection is intended to achieve a smoother and more symmetrical layout than would otherwise have been obtained with a layout performed in the lower dimension. In brief testing this has in fact proven to be the case and graphs projected onto two dimensions but generated using three dimensional space tend to be less prone to folding and possess a smoothness not otherwise possible.

The user may choose to adopt the naïve repositioning technique to improve performance of the layout process. This technique, elaborated on in the algorithmic discussion in this report, often gives equivalent results to the recommended repositioning technique, but with gains in the speed of execution of the implementation.

The two, primarily visual, options available are for enabling node colour-coding and the more extensive ‘eye-candy’ setting. In the former technique each node is assigned a colour, chosen from an even interpolation between green, orange and red, to indicate the energy at that node. This gives a readily interpreted indication of the energy content in certain parts of the graph and hence their stability. The ‘eye-candy’ function is an extension of the energy-based visualisation where each node is assigned a radius, for display purposes, proportional to the energy at that point. The colour of the edges between the nodes is also interpolated between the colours at each end-point. The effect of this is much more visual and more clearly represents artefacts in the energy surface of a graph than simple colour-coding, especially in a particularly dense and crowded layout. The effectiveness of both of these techniques is discussed in the evaluation of the product.

Scalability

As the proposed scalability of the algorithm in an ideal implementation was a major impetus for developing an implementation to test it in practice, a brief justification of this will be given. When ignoring the complicating factors affecting the theoretical performance of the algorithm, elaborated on in the algorithmic evaluation given, the implementation was found to give linear results. This merely satisfies the base requirements for the product but is an important factor to investigate due to the far-reaching affects a poorly scalable implementation would have on a critical analysis of the algorithm in practice. A theoretical analysis of the scalability will not be given as this will simply be a duplication of the scalability analysis for the original algorithm. Instead, a practical, ‘wall-clock’ analysis of the performance, with a fixed set of parameters, will be given below as a simple demonstration the scalability of the implementation. Note that maintaining the parameters fixed during the analysis is logically flawed since the performance metric should be the time taken to produce an acceptable layout for a certain size of graph, not, as is currently being investigated, simply the time taken to complete a run through the algorithm. This does not take into any account the quality of the results, and this fact is openly acknowledged. It is however this precise scenario in which the theoretical performance of the algorithm was proposed. The investigation here is simply to prove that the implementation possesses the same practical scalability as the theoretical scalability of the underlying algorithm. Again, flaws in the proposed theoretical scalability of the algorithm are addressed when evaluating the effectiveness of the algorithm in general, in this report.



From the above plot of execution time versus the number of nodes, the linear performance is readily apparent. Note that the average degree (the number of edges at each node) was kept constant at each measurement.

Development Process

Project Plan

To ensure the success of any software project of significant size, a thorough plan must be formulated to guide the progress of the work. Although a complete plan before beginning work is desirable, it is often feasible only to expand an initial, rather general, plan as milestones are completed. This was the approach taken in the development of the project and was adopted to ensure success through a certain degree of flexibility. Further safeguards to success were included to ensure the requirements were met, and extended upon, which will be discussed in concert with each other.

A very general set of project milestones were included with the project description given. These formed the basis for an expanded plan and are given as follows:

Prescribed Milestones

March 18th

- Project must be decided upon and in agreement with the project supervisor.

April 8th

- Basic requirements statement must be formed.

- Resources for the project must have been obtained.

May 3rd

- A progress report must be formed.
- A realistic plan for completion of the project is required.

May 31st

- The project should be finished.

June 14th

- A project report must be submitted.

These general guides are intended simply to ensure that individual students are not afforded the opportunity to fall behind in the development of their project. As such, it would be unwise to rely solely on these milestones to guide project development. To ensure a particularly successful project, rather than one that simply meets basic requirements, the following personal milestones were set during the very early stages of the project.

Personal Milestones

March 13th

- Nature of project should be established with supervisor.

March 24th

- A complete set of basic requirements for the project should be established.

March 29th

- Functional coding on the project should have begun.

May 5th

- The project should be complete according to initial specifications.

May 19th

- All extensions affecting the functionality of the program should be tested and complete.

May 26th

- Project should be complete, including any aesthetic extensions, and tested.
- Formulation of project report should commence.

June 14th

- All assessment material must be complete.

These milestones were elaborated on to form an initial plan for the project. This plan was kept to a general level to allow for flexibility in the project and the intention for an extended plan later in the project development

Initial Plan

Week 1

- Investigate the available projects with different research groups.

Week 2

- Decide upon project with a research group.
- Establish the exact nature of the task required.
- Research into area of project.

Week 3 – Task Fully Established

- Develop requirements of task.
- Formulate the remainder of the initial project plan.
- Begin structural plan for project.

Week 4 (including mid-semester break)

- Further investigate task and complete structural plan.
- Thorough investigation of algorithm used.
- Begin initial coding on project
- Develop a working (properly integrated) graph layout module for WilmaScope with nil functionality.

Week 5

- Extend layout engine with limited functionality using the fast layout algorithm.
- Incorporate equation for evaluating the energy (potential) at a node.

Week 6

- Implement density matrix component.

Week 7 – End of Initial Plan

- Add simple user controls for use of implementation.
- Evaluate progress up to this point and develop extended plan for the remainder of the project.

By the time actual coding of the project had begun in earnest, and the project was nearing a useable form, satisfying the basic requirements, an opportunity was provided to expand on the initial project plan. A plan of this detail was not possible during earlier stages of the project simply because it was not possible to predict the exact development tasks, while maintaining a fluid plan, capable of handling unexpected events. This expanded plan has the additional benefit of satisfying the requirements for the progress report required for submission. A listing of the plan developed at that stage is given below.

<u>Expanded Plan</u>

Week 8

- Test work to date.
- *Extension:* Implement mass and radius dependent circular attenuated node footprints.
- Use weighted centroid algorithm for barrier jumping.
- Make options panel to allow the user to vary program parameters at run-time.

Week 9 – Basic Requirements Met

- Present essentially complete (in base form) project to Tim, and get feedback.

- Integrate module with Wilma more tightly to allow for graphs opened from file, dynamic modification of graph elements etc.
- Begin analysis of ideal program parameters (will most probably opt to make minor modifications to the code as a result of this analysis).
- Further scalability analysis and testing.

Week 10

- *Extension:* Node colour-coding according to potential energy.
- Modify code to try to achieve faster execution and more efficient memory use.
- *Extension:* Implement consideration in algorithm for the natural length of edges, and an option for varying the 'springiness' of the edges.
- Generate some test graphs.

Week 11

- *Extension:* Node colour-coding according to potential energy.
- *Extension:* Utilisation of third dimension of node position.
- Testing and analysis of parameters.

Week 12 – Code Deadline

- Final testing of extensions.
- Further tuning of code in lieu of report and to aid in demonstration.

Week 13

- Write documents for Tim on how to create an alternate layout engine.
- Report and program analysis.

Week 14 – Report Deadline

- Report and program analysis.

Process

To illustrate the project work being completed either according to or ahead of plan, a description of the important work completed over the period of the project will be given. This will also make mention of the more important meetings and correspondence held, guiding the development process. It can be seen from the process given that the work done on the project was completed in many cases ahead of time, leaving opportunity for extensions to the basic requirements of the product.

The quality of the work completed is evidenced by both the usefulness of the implementation, and the resoundingly positive feedback from Tim on the project, discussed in the evaluation. While the quality of the product is evidenced by these factors, the quality of the process itself can be reckoned from the process description below and the evaluation of the process given later.

Summary of Tasks Completed

Week 1

- Looked at online descriptions of research groups[2] to find one that appealed to my interest.

- Emailed Professor Peter Eades about doing a project for the Information Visualisation Research Group
- Organised meeting with Tim Dwyer to discuss the possibility for a project.

Week 2

- *Meeting:* Met with Tim Dwyer for the first time to discuss the project he wants done.
- Began investigation into the scope of the project.
- Obtained approval from subject coordinator on the project.
- Informed Tim of the decision to stick with the project and discussed the requirements involved.

Week 3

- *Meeting:* Discussed project in more detail with Tim Dwyer.
    - Finalised program requirements.
    - Discussed the structural design of the package the module is to be written for.
- *Meeting:* Attended the group's weekly meeting, was introduced to the rest of the research staff and was able to view presentations on their research work.
- Obtained source code for WilmaScope package.
- Began an examination of the structure of the package source code.
- Developed the remainder of the initial plan for the project.

Week 4 (including mid-semester break)

- Read in detail the paper describing the algorithm to be implemented.
- Discussed the interface within the WilmaScope package further with Tim.
- Designed the initial structure of the implementation.
- Began coding of the structure of the implementation.
- Formalised the established requirements for the project, to meet prescribed milestone.
- Created a layout engine that did not move the nodes at all, but functioned correctly within the WilmaScope package.

Week 5

- Tested all work to date.
- Created test code to generate graphs for the layout engine.
- Implemented function for calculating the energy at a node, without taking into account the density at that node.
- Extended layout engine to move nodes randomly during each iteration, in an attempt to minimise their energy.

Week 6

- Developed and tested the density matrix and associated functions.

Week 7

- *Meeting:* Met with Tim Dwyer to discuss the use of the layout engine since developed from within the WilmaScope package.
- Finalised detailed plan for remainder of project.
- The code up to this point was tested debugged.
- The algorithm parameters were tuned initially to achieve better results in practice.
- Final adjustments were made to the maintaining of the density matrix.
- Test code was written to automatically generate graphs and run the algorithm on them.

Week 8

- *Meeting:* Discussed with Tim Dwyer:
  - His happiness with the current state of the project.
  - The reporting that can be done when the project is finished.
  - Methods for calculating weighted centroid for barrier jumping.
  - The completion of the coding at least within the next 3 weeks.
  - A compromise between the current complexity of the density implementation, and the time gained through a simplification to it.
  - A plan for demonstrating a working version of the product by the next week.
- Simulated annealing technique was implemented and tested.
- The program code was made cleaner and more modular.
- Investigations were made into scalability of the implementation.
- *Extension:* Mass and radius dependent node energy footprints, and the extension of the potential equation to accommodate these, were developed
- Further debugging and testing.
- Wrote progress report.
- Implemented barrier jumping.
- *Extension:* Begun work on run-time adjustment of parameters.
- Further extensions to the implementation.

Week 9

- *Meeting:* With Tim Dwyer:
  - Demonstrated working version of product.
  - Suggestion made by Tim to demonstrate the project at an upcoming research group meeting.
- *Extension:* Fully implemented run-time parameter adjustment.
- Improved user interface.
- *Extension:* Added node colour-coding.

- *Extension:* Added extensive energy visualisation, altering node radius to represent the energy at that node.
- Debugged and tested work to date.

Week 10

- Meeting: With Tim Dwyer:
    - Obtained other research papers for reference on alternative layout techniques.
    - Discussed weaknesses present in the algorithm used.
    - Booked a time for my research group presentation.
- Extension: Added option to project a graph onto two dimensions (at this stage only to make existing 3D graphs compatible with algorithm).
- Implemented use of line-drawing graph rendering mode.
- Tested the performance of algorithm in resolving grid-based graphs.

Week 11

- *Extension:* Modified implementation to operate in three dimensions if required.
- Extensive work on making the user interface more intuitive.
- Extension: Implemented naïve reposition technique.
- Tested and debugged work to date.
- Investigated the ability of the algorithm to resolve folds in a graph, particularly in three dimensions.

Week 12

- Final testing of code.
- Slight code-maintenance to improve readability and clarity.
- Documentation on project extended.
- Booked meeting with A/Prof. Alan Fekete to demonstrate the working project.
- Began serious work on report.

Week 13

- Critical analysis of algorithm and implementation in lieu of presentation for Information Visualisation Research Group.
- Planning of project demonstration.
- Significant reporting work done.

Week 14

- Demonstration of working project to A/Prof Alan Fekete.
- Wrote presentation for research group meeting.
- Delivered presentation on project and algorithm at meeting.
- Finalised report.

Final Presentation

On the 13th of June, I presented a talk to the Information Visualisation Research Group, The University of Sydney, on my project and insights garnered into weaknesses in the algorithm. As part of this presentation I also offered possible theoretical extensions to the algorithm that could be implemented to improve its effectiveness. As the nature of this project was not a research one, I did not develop these suggestions further as they would be the basis of a much greater theoretical work. A list of these suggestions and their justification is given as part of a critical analysis of the algorithm, in the evaluation towards the end of this report. A link to the abstract for my talk is given online[16], with the slides for the presentation available on request. For convenience, the abstract for the talk is given as follows:

Title: Implementation of Fast Force-Directed Graph Placement Algorithm

I will be presenting an implementation of a force directed placement algorithm for arranging very large graphs in linear time as discussed by Davidson et al, InfoVis2001. This implementation was written for the purposes of independently evaluating the feasibility of such an algorithm. Through the implementation I have discovered previously suspected weaknesses in the algorithm not acknowledged by the original proposal. The extensible WilmaScope 3D Graphing System environment was used to implement the algorithm, showcasing the ability of the system to extend to alternate layout algorithms.

This talk signified a debriefing as such on the project and afforded me an opportunity to demonstrate my work to the research group. The talk was given as a scheduled presentation at one of their weekly group meeting. Although any feedback given from this presentation would not been able to be implemented, due to the project being complete at the time of the presentation, it provided a valuable means for evaluating the project from an external viewpoint. In this case no negative feedback or suggestions for improvement was received on the project, due in part to my own discussion of the weaknesses and possible further direction to be followed. The feedback given focussed on the algorithm itself and served to reinforce my own initial evaluation as to the weaknesses apparent in the concept. As this feedback was all in agreement with my current thinking on the project, it served as a valuable reinforcement of my beliefs and facilitated a somewhat more critical analysis of the algorithm in this report than would otherwise been given considering my limited research capacity on the subject.

Safeguards

The plan described above was engineered to provide as much opportunity for independent work as possible. The reasons for this direction were many. Merely as a courtesy to the important work done by the researchers it allowed me to work on the project independently for a large proportion of the time without having to interrupt Tim's schedule for meetings to discuss issues that could have otherwise been avoided. It also allowed me to ability to continue with the project at my own pace, within the restrictions of the plan and milestones, without being limited by the tasks currently being discussed with the project supervisor.

From discussions with other students developing similar products, it was apparent that the requirement of some supervisors for weekly, or even more frequent, reporting on the state of the project actually proved a hindrance to project development. The extra time required to formulate regular reports on smaller issues impinges on the effort that could otherwise be spent on developing the project itself. It also means that the student is afforded little flexibility in terms of the rate of development they assume, dependent on their time available for that period. The concept of many 'hard-deadlines' for work means that quality is often compromised in aid of completing the task by the specified deadline when the student's workload is particularly high, while not allowing them free reign for extending themselves when afforded more time in a given week.

In the methodology adopted in this case, I feel that it increased my initial chance of success somewhat, while allowing for the possibility of a greater quality project. The flexibility afforded in more general milestones early on in the project, as well as an evolving plan, allows the student to accommodate for the many unexpected duties they are required to undertake, particularly in response to assessment in other subjects. It does however require a degree of dedication and self-motivation from the student, and as such is not an ideal solution for all projects. Here I tailored the plan to suit my own style of work, and to allow the flexibility required when undertaking a large project, without enforcing stringent weekly reports that would be required if a student lacked direction and motivation in their approach.

The three basic project plans, developed as the project progressed, afforded a degree of fluidity not otherwise attainable in a more rigid plan. These plans included the initial milestones, the plan for early development of the implementation, and the refined plan for not only completing the task but to ensure high quality in all aspects. Although no significant set-backs were encountered, this flexible plan would have allowed for easy scaling of the process to ensure completion by the due date, regardless of difficulties encountered.

The deadline for completing the project to initial requirements was set quite early to ensure for a long period before the submission data to act as a buffer in case of unexpected problems with the implementation. The time gained could then be used in furthering the extensions to the project. This effectively broke the project into two parts – that leading up to a base implementation, and the period spent enhancing and extending the implementation developed up to that point. As a matter of personal interest, the implementation of extensions to the base requirements was an issue of great importance, and this segregation of the project into two parts ensured adequate time was allowed for these extensions. If there were additional difficulties in the project not otherwise accommodated for in earlier buffers, there would still have been a long time remaining after this initial deadline to complete the task.

In all cases of milestone development, the work was completed well ahead of the assessment deadlines. Meetings were also booked with Tim Dwyer to demonstrate the implementation at each of these major developments to ensure the work was being done of high quality and to requirements within the time-frame. Discussion with Tim at these meetings on the plan for the next unit of work allowed the pre-empting of any possible difficulties that had newly arisen, and the inclusion of steps to combat these in future plans.

Evaluation

Product Evaluation

An examination of the strengths of the product will first be given before an analysis of its weaknesses and areas for improvements or further development. A discussion of the benefits of the project within the research group has already been given with the project context.

Strengths:

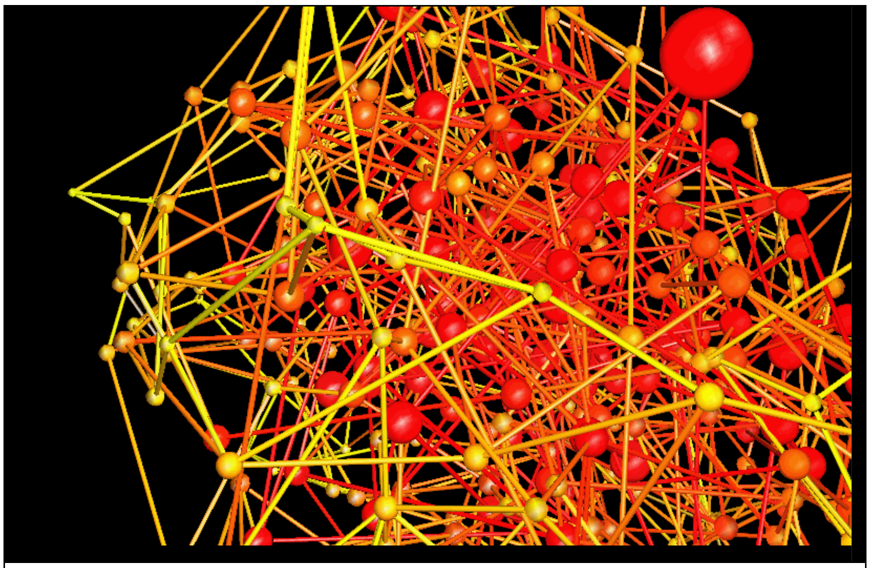
With the intended purpose of the implementation being to provide a means for evaluating the algorithm in practice, most of its strengths lie in its effectiveness in providing an environment tailored to the uses of a researcher into the algorithm.

If the implementation was to provide any degree of usefulness as a research tool it had to allow for easy modification of all parameters involved in the algorithm. This concept is taken to a further level in the project where the user is able to modify any parameter of the algorithm while it is actually running. When a parameter is modified at run-time via one of the sliders provided in the interface, the internal representation used by the algorithm changes dynamically and seamlessly to move to a state where the new parameters are being used. With this feature the user can begin the layout of a graph but make an educated choice to modify the parameters too see the effect on the current state of execution. As an example of the potential use of this feature, a user might notice that a graph has become unfolded and reduce the rate of barrier jumping to produce a more accurate layout. Alternatively they may see that the graph is not readily reaching a stable state and increase the number of iterations to run the algorithm for. In the latter case, the length of the phases (boiling, quenching or simmering) will be automatically scaled to produce results for the remaining iterations as close as possible to what they would have been if the iterations had been set to the higher value before the run.

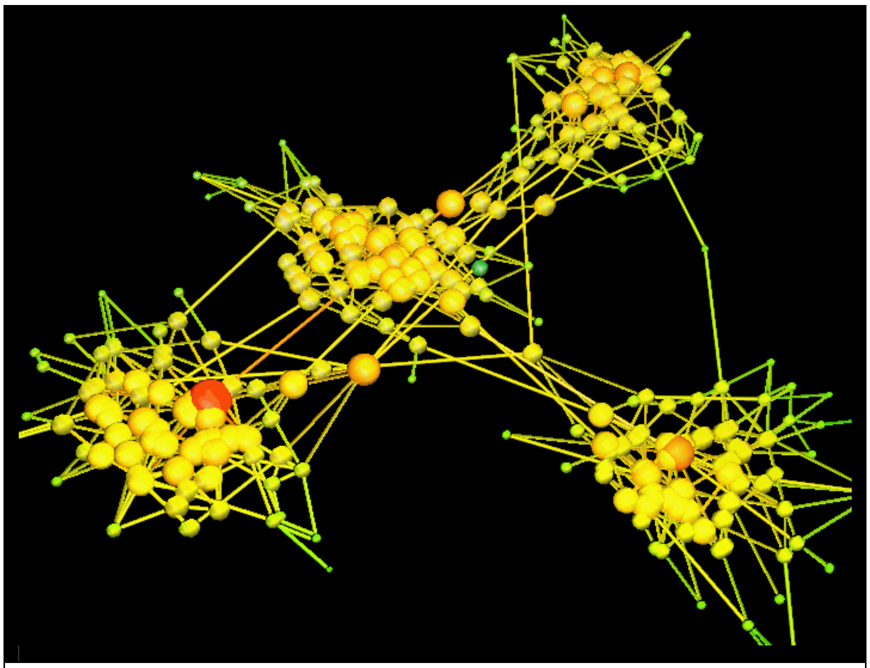
The expansion of the implementation into three dimensions, with the option for 2D or 3D layout available to the user, is a strength of the package with interesting consequences. Coupled with functionality to project a layout onto a lower dimension, this feature was the result of an unexpected insight. The implementation was increased to three dimensions purely for interest sake but when tested it was found that graphs had a much higher probability of unfolding themselves in three dimensions than in two. This is due to the fact that in three dimensions the graph is able to gradually unfurl, given a full three degrees of freedom, whereas in two dimensions a whole section of the graph must undergo a complete reflection to undo the effect of a fold. This result signposts a whole new area of research in approaching the generation of a graph layout. If a graph is laid out in a higher dimension (not necessarily restricted to three) and the projected onto a lower dimension, the results will be much smoother and less prone to anomalies produced by folding. It was since discovered through discussion with Tim Dwyer, that this approach has in fact already been taken in a new breed of algorithms, and that they function very well in practice. The limiting factor on this multidimensional technique would be the memory required to operate the algorithm and store the density of nodes in a higher dimensional space. A more thorough look at folding is given in the evaluation of the algorithm.

The modularity of the implementation is area in which it shows particular strength. The calculation of the potential at a node is abstracted to a single method, as well as the generation of a new position for each node. This allows extensive modification to the algorithm and the effect of the implementation while only modifying two methods. The most modular aspect of the implementation is that of the density matrix and associated functions. These are all abstracted into a class representing the ‘universe’ the nodes reside in. A request can be made to the universe object created from this class to determine the density at a particular point. The density class may be modified extensively, or replaced with an alternative density algorithm, without affecting the remainder of the implementation. This would be particularly useful if the choice was made to use an alternative method for calculating density, due to the relatively high memory expense of the density matrix currently used.

Among other features of the implementation, one which is particularly useful is that of the node-potential visualisation functions. To analyse the algorithm in practice fully, a researcher would need to pause the algorithm mid-execution and examine the energy of nodes at various points to determine how readily these nodes move in a direction minimising their energy. Two features have been implemented to present this information immediately and unambiguously to the user. The first of these is node colour-coding where the nodes are assigned a colour from green to orange to red, dependent on the amount of energy at that node. To provide an even more visual metric, particularly for large graphs, a feature is also provided to adjust the displayed radius of the node dynamically, according to the energy at that node. This is particularly useful in identifying clusters in a graph that has undergone the layout process. The user can scan the graph for larger bulges towards the centre of a cluster to indicate a high density and energy at that point. In analogy this is similar to the way a galaxy would appear more dense towards the centre, and how individual galaxies may be resolved by paying notice to the bright central core. The feature aims to provide a visualisation that is totally natural to the user and appeals to their logic in identifying clusters without any deep analysis of the graph. The following two images show two cases where this node-potential visualisation can be useful.



A graph layout in process – note the very high energy node in the top right corner



A very simple clustered graph – note the higher energy towards the centre of the clusters and the higher energy in the bottom left cluster

Weaknesses and Improvements:

Ignoring any weaknesses in the original algorithm, the main disadvantage of the implementation is the speed of execution it provides. In an attempt to fit in with the current work in the research group, the implementation was written in Java. The impetus for this was to provide a convenient and easily maintainable module to fit into the WilmaScope graph visualisation package, which is also written in Java. The downside to this approach is that the objects used take up very large amounts of memory compared to their C equivalents. The node objects themselves must be stored in the same format used in the package, for compatibility reasons, and hence must be Java objects. This however is not the source of major memory use which is in fact the density matrix used. This matrix is used to store double precision floating point numbers and can become very large in memory terms. Much of this memory use is due however to the algorithm itself, which will be discussed in the following section. To make an improvement to the memory use without modifying the algorithm, the density matrix could be made to store smaller data-types, such as a single precision floating point number.

Unfortunately it is not possible to reduce the memory requirements for the nodes themselves and maintain the same degree of integration with the visualisation package. As this integration was a prime requirement for the implementation, the option to use another method to store the nodes was not taken. It would perhaps be possible, with a great degree of effort, to store the nodes in some simplified form, using the WilmaScope package just for displaying the resulting graph. This however would be a waste of the environment the WilmaScope package provides for graph modelling and display. Since the aim for this implementation was ease of use for research, rather than an extremely efficient implementation compromised in useability, the choice was made to utilise the Java object structure for storing the nodes.

A point could be made that the implementation places more emphasis on the attractive display of the graphs, and the ability to modify parameters at run-time, rather than focussing on a simplified yet efficient program. This issue is really a design decision made when analysing the priorities for the project. As has been stated, the implementation was not intended to be the most efficient or fast available, but instead makes some compromises in aid of useability. If instead the goals of the implementation were to create a product that could be used in practice for extremely large graphs, an alternate approach would have been made.

Along with the memory requirements of the implementation, the speed of execution could no doubt be improved. Many improvements such as the absence of animation of the layout process would result in a much faster layout generation, however these compromise the effectiveness of the implementation as a research tool. One real improvement that could be made is the utilisation of the support in the WilmaScope package for using an external program to compute the layout of the graph. This feature is included to allow for the use of faster programming languages and techniques in calculating the graph layout. If this option was taken, the layout algorithm, including the density matrix, could be coded in a typically ‘fast’ language such as C or C++. The Fast Layout module in the WilmaScope package could then be made to communicate with this program, and use it to calculate the layout after each iteration.

The gains through such a technique are potentially quite high. Established optimisation techniques such as pointer arithmetic could be used in the C implementation of the layout engine, as well as optimisation during compilation, to produce an engine which will run much faster than the equivalent implementation coded in Java and executed using the Java Virtual Machine. The limiting factor on this speed advantage would be the time taken for the Java module to communicate all the node information to the layout engine and receive the new layout. A real disadvantage of this approach however would be simply in the readability and maintainability of the actual program code, which would have to be split across two platforms, the module and the engine, and make compromises in simplicity for efficiency sake. The main reason this technique was not implemented however was the added inconvenience and time taken in engineering such a system. Due to the option for modifying the layout engine parameters at run-time, a complicated API would have to have been developed to allow communication between the module and the engine regarding the modified parameters, while the engine simultaneously developed the layout and may possibly return positional information on the nodes. This flexibility, while accommodating a dual-platform implementation, would have been very complex to implement. If speed of execution over maintainability was a greater priority then this option would have seriously considered. Nonetheless, it is an area in which further development could be done to produce a similarly functioning implementation with faster execution time.

While not necessarily a simple improvement to the current implementation, if it was expanded to accommodate multiple different algorithms this would pose significant value to the research community. The modularity of the implementation would allow an alternative algorithm to be developed on the existing framework without the additional effort required to develop the first. This algorithm would be chosen from others also claiming to be able to calculate layouts for very large graphs[3, 4, 5, 6]. In this scenario, the user/researcher would be able to use the same interface to experiment with the execution of both algorithms, using both on the same initial random layout, with similar parameters. As both algorithms would be implemented on essentially the same framework, a valuable comparison would be able to be made between all aspects of the alternative algorithms. This information would be very valuable from a research perspective and could form the basis of a paper on the subject, particularly if a larger number of different algorithms were all implemented over the same platform. In this project the choice was made to develop the one implementation thoroughly instead of producing implementations of multiple algorithms with compromised quality. With extended time these alternative algorithms could be developed and used to perform very interesting comparisons.

Algorithmic Evaluation

It would be an error to implement an algorithm without giving consideration to the effectiveness of the algorithm in practice. In a situation such as this, the potential for use of the product is very much dependent on the quality of the underlying algorithm. It could be said that the implementation, however well written, will probably have little extended use beyond research purposes. This is because the results produced by the implementation point to a number of weaknesses in the algorithm, and suggest that other methods would be better for real-world use. This is a very interesting result from a perspective of product development as it shows a direct impact the program has had on research, in that it has provided a result affecting the path of further research. As a demonstration of the importance of the project within the research community, these weaknesses will be discussed as well as the aspects of the implementation indicating them.

A major criticism of the algorithm is that it is too ‘random’ in its approach. The method for moving a node consists of simply moving it around in totally random directions, returning the node to its original position if the move was unsuccessful. This can lead to a problem where a node needs to move in a certain direction to minimise its energy but all of the random directions are directed in different orientations. In this case it may take many iterations before the node moves in an ideal direction, with all of the previous iterations being wasteful. This entails a high degree of inefficiency and naivety that could otherwise be compensated against. It would in fact take only a slight increase in time to calculate a direction vector from the attraction forces of the neighbouring nodes. In its current form this directional information is essentially being ‘thrown away’ by the algorithm, a factor that implies that the algorithm is far from optimal as it does not make full use of the information available to it. A more advanced, and no doubt efficient, technique would be to use this direction vector to make a probabilistic decision as to the direction to move the node, within some degree of randomness. This would ensure that a much higher proportion of nodes are moved in a direction that decreases their energy, and reduce the time expended in correcting poor moves.

One of the major inflictions on the implementation of the algorithm is its high memory use, effectively prohibiting the use of large graphs. This in some part is a result of the predisposition to ballooning memory requirements possessed by the algorithm. The high memory requirements are caused by the necessity for a density matrix storing the entire ‘universe’ the graph resides in, and as is often the case, are the results of a compromise between speed of execution and memory use. The accuracy in calculating the density at a point in the graph is directly proportional to the resolution of the matrix. A higher resolution matrix is able to resolve finer detail in the densities at each point, however with sufficiently high resolution the matrix can become extremely large with an increase in graph size. As the algorithm was intended to for calculating the layout for very large graphs, this memory requirement can become an early limiting factor. The problem is exacerbated greatly in higher dimensions where the memory requirements for the density matrix scale with order:

$$\Theta\left(r^{d}\right)$$

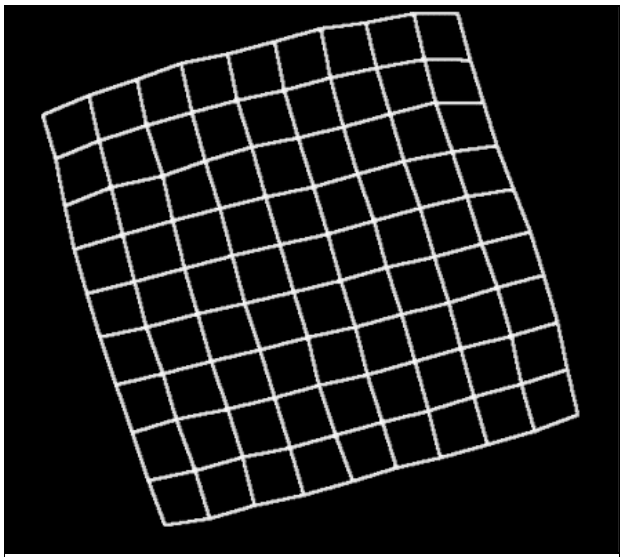
r = Number of discreet positions across radii in space

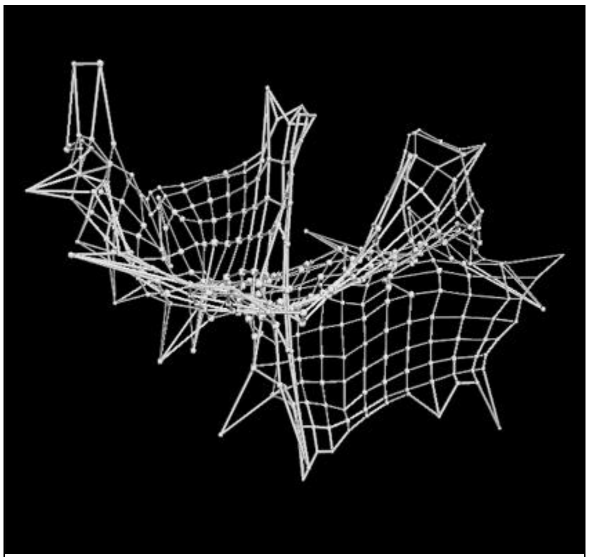
d = Dimension of the space

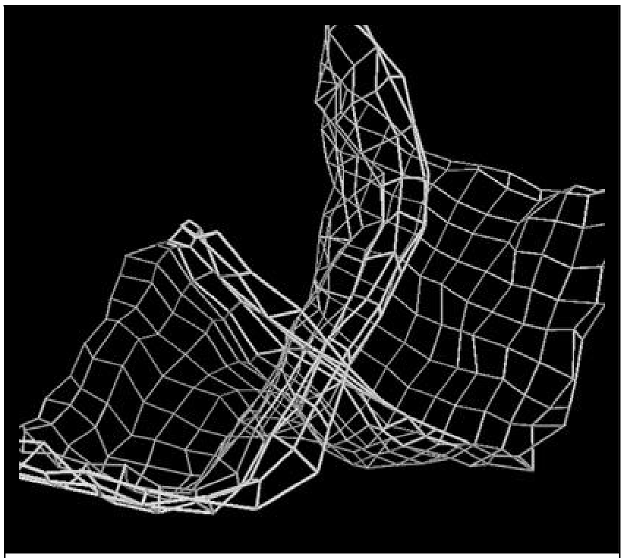
In my own experiments I found that memory restrictions limited the size of the graph laid out long before processing time became unacceptable. It was not possible, even with 300mb of available stack memory, to calculate layouts for two-dimensional graphs with nodes numbering in the high thousands, the number of nodes possible much lower for three dimensions. Note that the use of Java for the implementation resulted in much higher memory use per node, although the poor scalability of the memory required still remains.

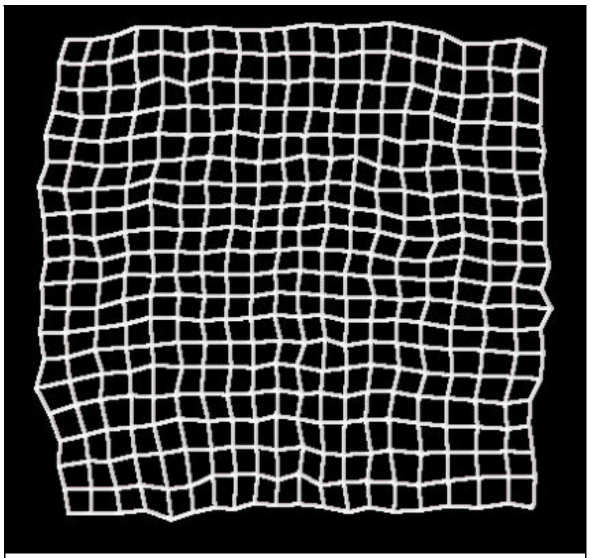
Folding

A problem plaguing algorithms of this class is that of ‘folding’ where the graph becomes hopelessly tangled due to an abundance of local minima in which nodes can become trapped. Although barrier jumping is implemented to avoid this occurrence, a single rate of barrier jumping is not sufficient to untangle all graphs. In fact, to be able to correctly lay out larger graphs, the parameters of the algorithm must be adjusted to provide higher resolution, a greater rate of barrier jumping, and more iterations. As an example of the relation between the parameterisation and the effects of folding in the resultant graph, as well as the requirement of higher resolution for larger graphs, a number of sample layouts produced by the implementation will be given. Note that the term “fine” in relation to the parameters indicates a higher resolution, greater rate of barrier jumping, increased iterations, and as such, greater memory requirements.


10x10 Grid – Accurate Results


14x14 Grid – Same parameters, evidence of folding


20x20 Grid – Finer parameters, still folding present


20x20 Grid – Yet finer parameters, no evidence of folding

It can be readily seen above that acceptable results for larger graphs come at the cost of the requirement for finer parameters. Despite the algorithm being linear given a constant set of parameters, the requirement to increase the detail for larger graphs effectively destroys the time bounds for the algorithm. In this situation the algorithm becomes much less deterministic, however it can be easily understood that given an additional increase in the detail required, the algorithm will perform at worse than linear scalability in practice. This is not an insignificant point, and to some degree, invalidates the goals of the algorithm to produce accurate results for very large graphs in linear time.

Process Evaluation

The process of such a project is primarily geared towards completing the project on time at to the requirements. The process also carries with it the extra requirement in that it must aid in the personal development of the student, and provide a valuable learning experience. As this project was undertaken primarily for educational means, the process must also allow time to reflect on the work being done and to explore other educational avenues. The following description discusses the effectiveness of the process in delivering the product, and also providing an educational experience.

Strengths:

The main strength possessed by the process followed was in the flexibility it provided and the safeguards against failure. The evolving plan was one concept adopted to help aid in this flexibility. To avoid restricting the progress of the project, the project plan was refined over the duration of the process. Initially a general scheme of milestones was established, while an initial investigation into the details of the product was undertaken. This ensured that the project would not be able to fall behind schedule to a significant enough degree to prevent a successful product resulting. Often when a detailed plan is completed before any actual work is commenced on the project, the latter phases of the plan are somewhat arbitrary and lack the insight gained from the early experience in the development. In this system I was able to avoid setting the plan for the minor details of the latter stage of the implementation until nearer towards the actual adoption of this plan. This provided me with a degree of fluidity that enabled me to adjust the capacity for extensions to the project in response to the time taken to complete the earlier stages.

Once the scope of the project was realised, a plan for the bringing the project to an ‘acceptable’ state was formulated. This allowed an additional sense of direction, avoiding the pit-falls that can arise in a project that is flexible to the degree that it does not assist in the project development. Nearing the end of this phase of development, a comprehensive plan for the final phase was written. The act of writing the plan at this late stage enabled me to capitalise on the time gained in the initial phases, and to incorporate enhancements to the project in the plan with the assurance that they will be feasible for completion. If this plan was written at the very start of the project there would be no way of pre-empting the subtleties of the earlier development of the project, and how they affected the plan for the rest of the implementation.

Another significant aspect of the process, from a personal, educational viewpoint, was the incorporation of time to investigate the effectiveness of the algorithm, as well as some brief research into other possibilities for algorithms to implement and test. Throughout the process, time was spent investigating the nature of the work done, rather than blindly implementing techniques. This led to a more educated project that is beneficial in two regards. It resulted in a project that more closely satisfied the original requirements, and extended on these in useful ways, because the intent for implementing the algorithm was fully understood. It also provided a valuable education beyond the scope of the project, into the nature of the field of research. This additional knowledge would allow me to improve not only the implementation, but also the underlying algorithm, in a future project.

Weaknesses and Improvements:

A possible weakness in the process followed was the comparative lack of rigidity in the plan. The use of an evolving set of plans, while ensuring the project was completed, allows the opportunity for scaling back the project due to a lack of work in earlier stages. This would particularly be a problem for a student who lacked self-discipline in their work. In this case it was not a negative issue and in fact proved to be a benefit, although this is very much dependent on the student following the plan. For this reason I would be reluctant to recommend such a strategy as a blanket technique for all students attempting such a task. A more generally applicable and safer option would be to make an initial investigation into the scope of the project, then formulating a plan to encompass the entire duration of the task.

It would have been desirable to attend every research group meeting for the duration of the project. This would have provided a greater knowledge of the happenings in the group. It would have also furthered the depth of information gained while working with the group. At these meetings it would have been possible to liaise with the project supervisor, Tim Dwyer, on the current status of the project. This would have also provided an opportunity for discussion with Tim without the requirement of organising a meeting and as such occupying more of his time. Unfortunately, due to conflicts with laboratory times for other subjects, attendance at all of these meeting was not possible. Instead, a minimum of these meetings were attended to gain an insight into the context of such gatherings, and also to present my own work to the group. In an ideal situation all of these meetings would have been attended, adding a new dimension to the development process of the project.

Although a presentation of the project was given to the research group, it would have been advantageous to present this work much earlier to be able to utilise suggestions from the group on the project. When the project was finally demonstrated there were in fact no suggestions as to improvements not already undertaken, and in this regard there was no loss in the timing of the presentation. If a partially complete project was demonstrated to the group at an earlier stage it would nevertheless alleviated some of the responsibility of performing my own critical evaluation. This would have lead to an easier project development, but not necessarily one which was more useful as an educational experience. In a case where the aim of the project is purely the end result and not the learning experience of the development, then it would be highly recommended to present a partially complete work at such a meeting.

In this case the project was not demonstrated early partially because of my own reluctance to demonstrate an incomplete, and in my own view, as yet imperfect, product to a research group more accustomed to viewing presentations on the result of PhD work. The logistics of organising a presentation when Prof. Peter Eades, the head of the research group, would be available for attendance, also affected the date of the final presentation. As a result the presentation had to be given after the project had been completed and submitted.

A minor weakness in the process was the lack of records kept on the first few weeks of project development. At this stage the plan for the rest of the project was still being formulated and as such effort was devoted to this rather than the logging of activities completed during that time. This did not have any impact on the completion of the project but instead made the reporting on those first couple of weeks slightly more difficult. All this information was available through email logs but it was not collated and presented clearly like the process of the remaining weeks of the project. In future a log will be kept of *every* main task completed for reference during reporting.

A final weakness when reflecting on the process was that the completion of the report was planned, and completed, very close to the due date. Time before this reporting was spent making final checks of the software product and enhancing the clarity of the program code itself. Although this did not raise issues for the actual completion of the report, it meant that a number of opportunities were lost in relation to its formulation. Nearer to the submission date offers were made by Tim Dwyer and others to review the report and offer suggestions as to how to improve it. Unfortunately, due to the progress of the report at that stage, a significant enough draft of the report was not available for review. As the possibility of review by a more experienced person had not arisen in my previous studies it was no something I pre-empted and accommodated for. This has been a learning experience however and in future an initial draft copy of the report will be written well ahead of time, to be evaluated and extended upon according to suggestions from other parties. This feedback would no doubt have impacted on the eventual style of the document and would have eased the final development of the report.

Learning Experience

The most general learning experience partaken in through this project has been that of participating in the development of a project of scale not previously attempted. This opportunity for developing a practical project for real use by a group has provided an excellent grounding for future work in industry. Time-management techniques, requirements analysis, and communication skills have all had to be utilised in the development of the project. The fact that the project was attempted alone, for the needs of a 'customer', provides skills in self-motivation and independent work outside the educational setting. While earlier endeavours in my educational career have helped build my knowledge and experience, this project has allowed me an avenue to combine these skills in producing a practical software package. It adds a vocational aspect to the latter stages of an education which has often been traditional in its approach. I feel that the completion of this project has allowed me real vocational experience in addition to the benefits and capacity for reasoning imparted by much of the traditional scientific education and university experience provided by The University of Sydney. It is also a culmination of the Problem-Based Learning techniques pioneered within the department to provide a gradual transition from the education gained and the application of this education in a practical setting.

While I don't feel that my specific programming skills have improved to any significant extent through the project, the capacity for application of these skills has certainly grown. This, I believe, is what separates an individual who is simply good at programming, from one who is able to achieve useful and important results from their programming skill. The study of this subject, and subsequent project completed, has helped me learn to use my skills in a more practical setting, and become a more appealing prospect for an employer as a result.

The most valuable experience gained from this project however has undoubtedly been the immersion in a research field, and the exposure gained to real researchers and their work. Although the project did not involve any specific research, the context of the project itself has provided a valuable learning experience. The opportunity to work with Information Visualisation researchers, attend research group meetings, and eventually present a talk at such a meeting, is one that would be very difficult to attain if not for the nature of the project.

Before commencing this project, I did not have an intimate knowledge of the nature of research work, nor of the daily interactions of a researcher. The project has allowed me to attend meetings with Tim Dwyer, the researcher whom I coordinated my project with, where I have been able to see the results of research, interact with researchers, and experience aspects as simple as the office in which they work in. This has given me a taste of work in the research community that I would otherwise had not had, and prepares me a great deal for future honours work.

Through attendance at research meetings I have been able to witness the efforts of other researchers, and garner an appreciation for the level and scope of work involved. This culminated in a presentation made by myself to the group, on my project and, perhaps more importantly, my critical evaluation of the algorithm. The experience gained now, at this early stage of my academic career, will no doubt ease the transition into later research and provide me with an advantage due to the comfort gained in such an environment.

Of particular interest to myself is the scope provided by the project in improving my standing as a 'computer scientist' rather than a adopting a naïve view of the project merely providing benefits of a vocational nature. I have been able to make critical analysis of the algorithms implemented, and develop in a research capacity. Although this project did not entail any specific research, it has allowed me to make an evaluation of an algorithm proposed in a real research setting, for my own personal benefit and interest.

It is hoped that the contacts gained through this work will prove useful when seeking research work in the future. At the very least the project has provided me with a foundation in research work and suggests further avenues to explore in a research environment. The experience gained with the Information Visualisation Research Group has been a positive one and, if given the opportunity, poses a real possibility for work.

Conclusion

With most of the concluding remarks made in the final evaluation of the project and process, especially in the examination of the learning experience, this opportunity will be taken to summarise the net result of the project. While both strengths and weaknesses in the product produced, and process followed, have been examined, an overall conclusion on the depth of the project has not been made.

The project undertaken, and the assessment involved, has proven beneficial in a great variety of ways. While the lack of lectures has removed the direct imparting of knowledge from teacher to student, the experience gained through this project has been much more valuable. In providing a means for assessment where a student is afforded the flexibility to undertake their own project, the course acknowledges the experience gained by students during their earlier formative education, and instead provides an arena for utilising that experience. The benefits of such a task have been wide-ranging – improving both a students technical knowledge, but also their level of confidence and experience in undertaking a legitimate software development task.

It is felt that the experience of this project will prepare me for future research and software development in a way not previously provided in my education. For this reason, despite the obvious success of the project as a research tool, the experience of the subject as a whole can be seen as even more successful and worthwhile.

References

1. "Mission Statement", http://www.cs.usyd.edu.au/~visual/, Information Visualisation Research Group, School of Information Technologies, The University of Sydney, Sydney, Australia, 2002.
2. "Research Handbook", http://www.it.usyd.edu.au/research/research_hb.html#IVG, School of Information Technologies, The University of Sydney, Sydney, Australia, 2002.
3. Harel, D. and Koren, Y. "A Fast Multi-Scale Method for Drawing Large Graphs", Technical Report CS99-21, Dept of Applied Mathematics and Computer Science, Weizmann Institute of Science, Israel, 1999.
4. Gajer, P., Goodrich, T., et al., "A Multi-dimensional Approach to Force-Directed Layouts of Large Graphs". Department of Computer Science, John Hopkins University, Baltimore, U.S.A., 2001.
5. Walshaw, C. "A Multilevel Algorithm for Force-Directed Graph Drawing". School of Computing and Mathematical Sciences, University of Greenwich, London, U.K., 2001.

6. Quigley, A. and Eades, P. “FADE: Graph Drawing, Clustering, and Visual Abstraction”, Department of Computer Science and Software Engineering, University of Newcastle, Callaghan, Australia, 2001.
7. Davidson, G., Wylie, B., et al., “Cluster Stability and the Use of Noise in Interpretation of Clustering”, Sandia National Laboratories, U.S.A., 2001.
8. Fruchtermann, T. and Rheingold, E. “Graph drawing by force-directed placement”, Technical Report UIUCDCS-R-90-1609, Computer Science, Univ. Illinois, Urbana-Champagne, U.S.A., 1990.
9. Eades. P., “A heuristic for graph drawing”, Congressus Numerantium, 42, 1984, 149-160.
10. Wilcox, R., “Introduction to Robust Estimation and Hypothesis Testing”, Academic Press, 1997
11. Kamada, T. and Kawai, S. “An Algorithm for Drawing General Undirected Graphs”, Information Processing Letters 21, 1989.
12. “Group Members”, http://www.cs.usyd.edu.au/~visual/people.html, Information Visualisation Research Group, School of Information Technologies, The University of Sydney, Sydney, Australia, 2002.
13. “Capital Markets CRC Limited”, http://www.cmcrc.com/, Capital Markets CRC Limited, Australia, 2002.
14. “WilmaScope 3D Graph Visualisation System”, http://www.wilmascope.org/, WilmaScope.org, Australia, 2002.
15. “Visualisation”, http://www.cmcrc.com/cmcrc_files/vis.htm, Capital Markets CRC Limited, Australia, 2002.
16. “Events”, http://www.cs.usyd.edu.au/~visual/june02.html, Information Visualisation Research Group, School of Information Technologies, The University of Sydney, Sydney, Australia, 2002.