Movement as an Aid to Understanding Graphs

John Bovey, Peter Rodgers, Florence Benoy
University of Kent, UK

J.D.Bovey@kent.ac.uk, P.J.Rodgers@kent.ac.uk, P.M.Benoy@kent.ac.uk

Abstract

This paper describes a graph visualization method that attempts to aid the understanding of graphs by adding continuous local movement to graph diagrams. The paper includes a discussion of some of the many different kinds of potential graph movement and then describes an empirical trial that was conducted to investigate whether one kind of movement helps with a particular graph comprehension task. Although the results of the trial are promising, the degree of benefit afforded by the movement varies between graphs and the paper includes a discussion about graph features which may account for this discrepancy.

1: Introduction

Graphs are widely used as a visualization technique for interconnected data. Application areas include network mapping, software engineering diagrams and social network investigation. However, the layout of nodes and edges on the screen is very important for understanding the data and in many cases automatic layout can produce untidy layout where node and edge occlusion appear. In this paper we describe a technique that seeks to make graph diagrams more understandable by introducing small movements of the nodes about their location on the screen.

There are two major reasons why a moving graph diagram may be better than a still diagram. The first, and most obvious reason is that any sort of continuous motion should make it easier to resolve occluded detail. For example, given the node and edges in Figure 1, it is not clear whether there are two horizontal edges that meet the node or whether the node happens to lie on top of a single horizontal edge. If the node and edges were wobbling, even slightly, then it would be obvious whether the node and edges are joined since they would move together rather than independently. There are other kinds of occlusion that should be easier to resolve in a moving graph. For example, a pair of edges that lie on top of each other or nearly so will be separated at least some of the time if the diagram is moving.

The second reason why a moving graph diagram may be clearer than a still one is that people are good at extracting information from the relative motion of things in the environment. This makes it seem possible that a moving graph should be easier to interpret than a still graph as long as the motion reflects the structure in some way. In other words, a user looking at the moving diagram may be able to get a clearer idea of the overall structure of the graph and relationships between its parts than could be obtained from a still diagram.

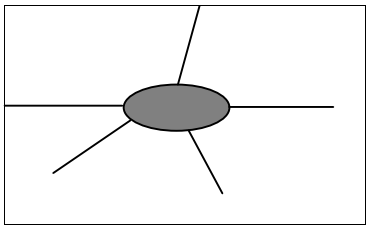

Figure 1

In order to investigate the usefulness of graph movement we developed an experimental software system that could be used to display graphs moving in different ways. This software was then used to run an empirical test with a number of undergraduate students. Our requirements for a simple controlled experiment means that these tests differ from how such movement might be used in a real world graph visualization system. In particular, any real application should allow the user full control of when the movement occurs, whereas the experiments simply presented the subjects with moving or still graphs. Our interpretation of the experimental results indicates the effectiveness of the system is very sensitive to the graph layout, and appears to be most effective when occlusion is present or the investigation of the graph has to cover a wide area.

We have not seen any previous work using this sort of small movement to improve the understanding of two-dimensional graphs. However, movement of various alternative forms has been applied to graph visualizations. A common technique is to animate the movement of nodes to illustrate the operation of graph algorithms primarily as an aid to teaching [1,3,5]. Another application of graph animation is designed to help maintain the user's mental map of the graph when comparing different drawings of the same topological graph. Here the graph is smoothly transformed between layouts [2]. Previous work in adding motion to aid understanding of a graph's structure includes [6], which describes experiments that involve rotating a three dimensional representation of a graph for comprehension.

The remainder of this paper is structured as follows: Section 2 discusses the varieties of possible movement that may help understanding a graph; Section 3 describes an empirical trial intended to discover whether movement is genuinely useful; Section 4 has an analysis of the experimental results and Section 5 contains our conclusions and possible further work.

2: Kinds of Graph Movement

Having decided that there may be benefits from displaying a graph by moving it slightly, the next question is what kind of movement will work best. There are actually quite a lot of different ways that a graph can be made to move – the motion could be random or periodic, large or small, fast or slow, and it can be distributed through the diagram in many different ways. In this section we discuss some of the alternatives, though there must be plenty of other possibilities that we have not thought of. The techniques we discuss here are: Random Movement, Circular Movement, Inelastic Movement and Movement as Annotation.

Random Movement: One simple and relatively easy way to add motion to a graph diagram is to let each node move in a random walk – this is easy to implement by repeatedly adding a random x or y increment to each node position and then simply redrawing the graph. We would probably want to restrict each node to stay close to its original position since, for movement to be truly useful as a graph display tool, it needs to work on graphs that have already been laid out using a static graph layout algorithm.

A variant on random walk can be much smoother. Here the acceleration of a node is modified rather than velocity directly. In this case the system must ensure that the velocity does not get too great. In addition, there is no direct way to completely bound the movement of a node, as the direction is only indirectly set by altering the current acceleration.

Circular Movement: Another possibility is to let each node move in small circles. Although we could let different nodes move in different sized circles at different speeds, a simple, and surprisingly rich, approach is to move each node in circles of the same size and period but at different phases. Two nodes that are 180 degrees out of phase will have large relative movement whereas nodes that have a small phase difference will move more nearly together. If we take this approach then we need some way to allocate phase angles to nodes – there are several different approaches:

1. Spread the phase angles locally within the graph so that each node has as much movement as possible relative to its neighbours.
2. Minimise local movement by giving similar phase angles to neighbouring nodes. The idea of this is to make the amount of relative motion of nodes increase with their distance apart.
3. Build the allocation of phase angles into the layout algorithm. This approach would require a layout algorithm that allocates to each node, not just the x and y coordinates but also a phase angle between 0 and 360 degrees. You could think of this as laying out the graph in a three dimensional cylindrical space consisting of the Cartesian product of two unit intervals and the unit circle.

Of these alternatives, 1 should give the best chance of uncovering occlusion, particularly by nodes or edges that are close in the graph. By making local structure move together, 2 or 3 should have a better chance of allowing the movement to bring out the overall structure of the graph but it needs more investigation to determine whether this really works or not.

Inelastic Movement: The motivation for this variant is that in both the random and the circular motions described above the nodes move independently and the edges are redrawn between the repositioned nodes, which is not the kind of motion that can be seen in any real world structure. If we are really trying to take advantage of an inherent human ability to understand the structure of things by seeing them in motion, then it might be worth looking at ways to make graphs move as if they were real life objects. One natural way to do this is to lay out the graph in three dimensions and then display a two dimensional projection of the moving, three-dimensional, layout [6]. An alternative approach that may work for sparse graphs is to lay out the graph in two dimensions and then attempt to move it in a way that maintains the lengths of the edges. This would give the effect of a collection of linked, inelastic rods being gently shaken. Modelling the graph's edges as inelastic rods would not be any use for dense graphs since they would form rigid structures but it may be possible to get realistic looking motion by introducing some elasticity into the edges.

Movement as Annotation: When graphs are used as models, the systems they are modelling generally have far more structure than simple nodes and edges. For example, the nodes may represent more than one kind of object and the edges more than one kind of link. Also, the objects represented may have numerical properties like size and strength. In a graph diagram this additional information is represented by using colour, differing node shapes, textual annotation and so on. It may also be possible to use movement to show some of this additional information. For example, if the nodes were of a number of distinct types then it would be possible to make nodes of the same type move in the same way. If the nodes represent objects of different sizes then the those that represent large objects could vibrate more slowly than those representing small objects.

3: The Experimental Environment

In this section we describe the software and test graphs that were developed to demonstrate the movement method we thought showed most promise. The software was designed to allow empirical experiments, so that the effectiveness of the technique could be explored.

To test whether adding movement to simple graph diagrams makes them easier to understand, we needed a task or tasks for empirical subjects to complete. Such a task should be one that is easy to explain to a student who might not be familiar with graphs, that involves looking at the graph in detail and that can be done reasonably quickly. There are a few such tasks that have been used in the past in empirical graph-interpretation experiments [4] but the one that we decided to use is that of finding the shortest path between two given nodes. This is certainly easy to explain and is easy to test on an interactive graph display – the test subject can simply be asked to click on the nodes that form the path. The drawback is that, unless the graphs are carefully chosen, it may not be at all clear to the subject whether the path they have chosen is the shortest or not. In principle, this should not be a problem since the number of correct paths is as much a test result as the time taken, but it does make it hard to compare times if a very large proportion of selected paths are incorrect. Our solution to this was to choose graphs and end nodes so that the shortest paths are always four edges long and are unique.

Our choice of graphs and paths was further constrained by the need to complete each student's series of tests in a one hour test session. We initially did some informal trials to determine what size of graph and length of path seemed to work best and we eventually decided that 5 node paths in graphs with about 16 nodes and 24 edges seemed the most appropriate. Using paths with 5 nodes meant that the test subject had to find three internal nodes to complete the path. Paths that are longer than this take a lot more time to discover, and shorter paths are rather too easy to find using trial and error. We originally intended to generate graphs with random topologies but, again after some informal experiments, the best approach seemed to be to use the same graph (shown in Figure 2, with the path in thick lines) for all the tests but give it different random layouts. That is, each graph diagram was created by taking the graph in Figure 2 and allocating a random position to each node. Some of the resulting layouts can be seen in Figures 3 to 6. In practice, the fact that all of the test graphs were the same was well hidden by the random layouts and none of the test subjects realised. It is, though, a feature of this particular path in this graph that none of the path nodes lies on a triangle, whereas all the nodes in the graph that are not on the path do form a triangle with two other nodes in the graph. In order to make the tests more uniform, we explained this fact to the test subjects before they started, so the trials can also be viewed as a test of the relative difficulty of finding triangles in still and moving graphs.

All the graphs, whether still or moving, were displayed in a 600 by 600 pixel display area with a dark background. The unselected nodes were drawn as red 9 by 9 pixel squares and the edges as single pixel lines in green. The highlighted path endpoints were distinguished by being coloured white rather than red. The test subjects were required to select, by pointing and clicking, the three nodes that complete the path between the two highlighted nodes. Nodes selected in this way were shown in yellow rather than red and the edges between selected nodes were highlighted by being coloured white rather than green. Selected nodes that turned out to be wrong could be deselected just by clicking on them again.

The goal of the trial was simply to attempt to determine whether movement is useful or not, so we had to make some decisions about the kind of movement to use. After some informal experiments we decided to use circular motion with a circumference of 20 pixels and a diameter of 8 pixels. The speed of movement was such that the nodes completed one revolution every 800 milliseconds. This is a level of movement that is sufficient to resolve occlusion but is not so violent as to make the graph uncomfortable to view. The nodes were made to move relative to each other by giving them different phase angles. We used eight different phase angles that were allocated to nodes using a simple depth-first traversal algorithm that tried to give different phase angles to neighbouring nodes.

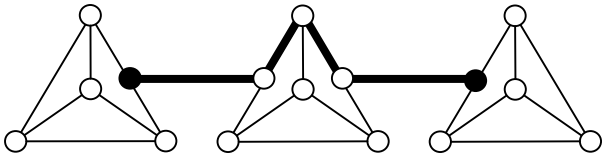

Figure 2

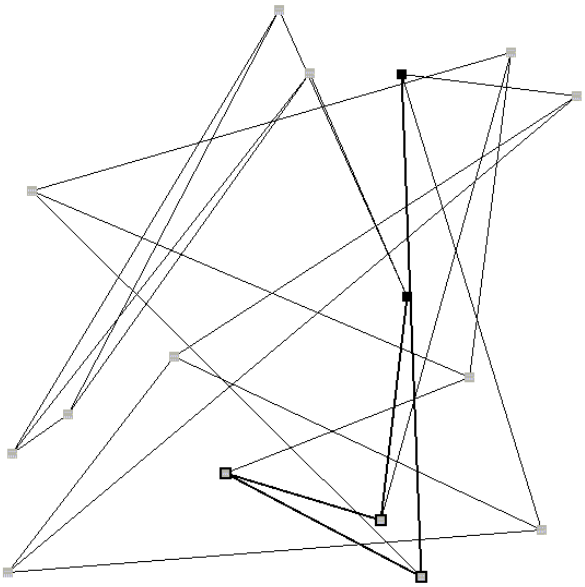

Figure 3: Graph 5

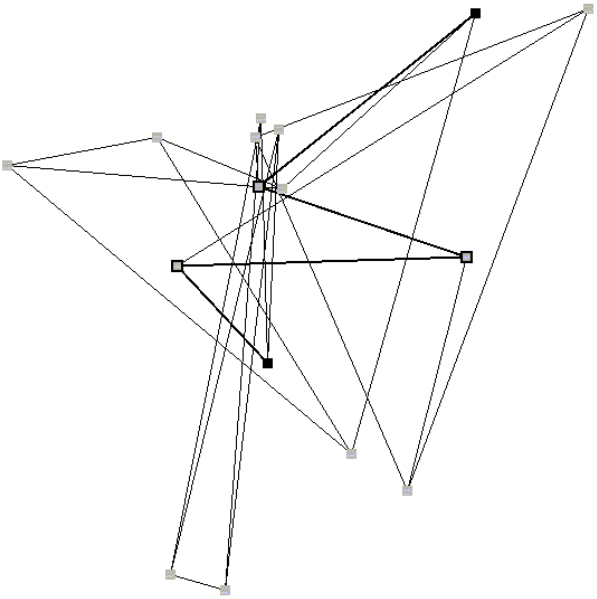

Figure 4: Graph 13

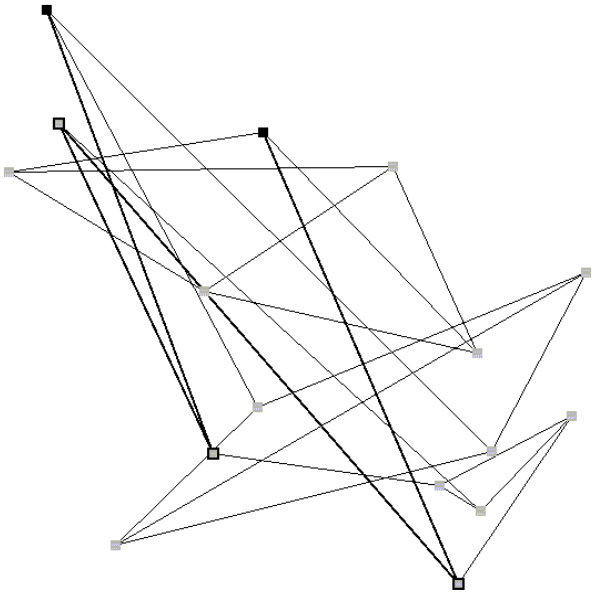

Figure 5: Graph 14

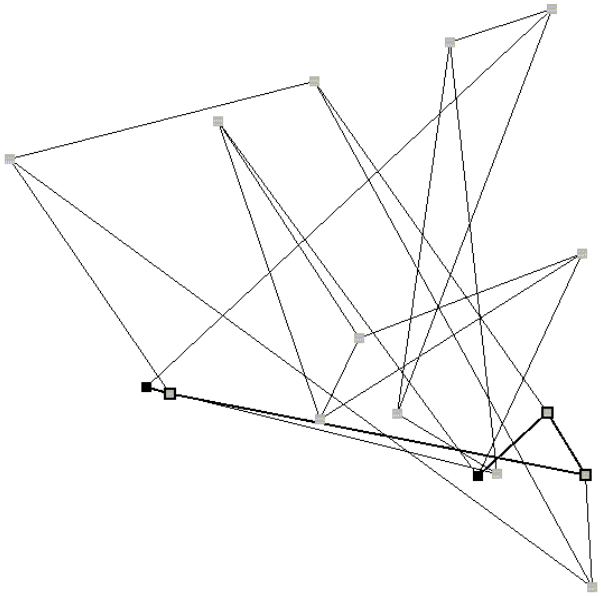

Figure6: Graph 18

In terms of organizing the trials we used 18 different graphs for the trials and every subject worked through a sequence that included each graph both moving and still. In addition to these sequences of 36 path-finding tasks that were timed and checked for accuracy, we also included 6 practice graphs that were added to the start of the sequences and were not included in the results of the trial. In order to minimise any effects due to fatigue, we divided the sequences into blocks of 6, each with alternating still and moving graphs, and provided rest pauses between the blocks.

Since each sequence contains each graph twice, once moving and once still, we wanted to minimise any bias that might be caused by subjects remembering the graphs and completing the path more quickly the second time. In part we did this by using graph sequences in which the moving and still occurrences of each graph were well separated, but we also used matching pairs of sequences with the graphs in the same order but the moving/still attribute exchanged. This means that for each subject who saw a particular graph first moving then still, there was another subject who saw the same graph first still then moving. We also used sequences in which the blocks of graphs were permuted so that different subjects saw different graphs early and late in their sequences.

4: Analysis of the Results

Our trials were done using 18 2nd and 3rd year Computer Science Students from the University of Kent (it is a coincidence that the number of graphs is the same as the number of subjects). The subjects were remunerated for their time and, since accuracy was important, they were offered a gift token as incentive for the greatest accuracy score.

Since each subject looked for the shortest path in each of the 18 graphs, both moving and still, that gives 18^2 or 324 pairs of trials that can potentially be compared for both accuracy and speed.

Accuracy irrespective of time: Of the 324 pairs, the 47 pairs that were both incorrect were discarded as nothing can be deduced from them. Of the remaining 277 pairs, 253 included a correct solution for a moving graph and 233 a correct solution for a still graph, which gives a small lead of 20 for the moving graphs. However, if the data is displayed by graph, that is moving versus still, there is a clear gain for moving graphs over still ones as in 13 out of the 18 graphs there were more correct solutions for the moving graphs, in only 2 graphs were there more correct solutions for still graphs and in just 3 graphs there were equal numbers of correct graphs for both moving and still. See Figure 7.

When time is considered as well, again the performance with moving graphs is, on the whole, better. Just over half (10) of the graphs have better average times than those for the still graphs, but it is interesting to note that the substantial differences in time are predominantly evident in the case where performance for the moving graph is better than for the still. See Figure 8.

If understanding of the graphs is gauged by considering, for each subject, the difference between the number of correct solutions over moving graphs and still graphs (moving minus still), 10 subjects performed better with the moving graphs, two performed better with the still graphs and for the remaining 6 there was no difference. See Figure 9. On the other hand if the average times are considered, for correct solutions, 12 of the subjects performed better with the moving graphs, see Figure 10.

Accuracy and Time, Statistical Analysis: Since each subject is presented with each graph, both moving and still, we can consider the data as paired, with each subject being his own control. Of the 324 pairs, 209 had the correct solution for both the moving and the still graph. Since it is difficult to assess the cause of failure to identify the correct path, particularly if the time for the incorrect solution is less than that for the correct one, the statistical analysis was carried out over these 209 pairs. The outcome of the t-test for paired data indicates that there are no grounds for supposing that the moving graphs were easier to understand than the still graphs. The relatively large standard deviation for the difference between the average times for each graph is indicative of the presence of some large mean differences over average times which are predominantly in favour of some of the moving graphs, see Figure 8.

An analysis of variance indicated a significant between graph effect that, coupled with the average time differences above, prompted a closer look at the graphs themselves. Close inspection of the times and correctness scores for individual graphs did not reveal any conclusive general evidence. This is not, perhaps, surprising given the small data sets for both graphs and subjects and the fact that the graph displays were generated randomly.

Questionnaire: As an aid to finding the shortest path, subjects were advised that if a node was part of a triangle then it would not lie on the shortest path. Subsequently subjects were asked whether they thought the movement helped in identifying either the triangles or the shortest path, 8 and 9 subjects, respectively, of the 18 thought the movement was helpful.

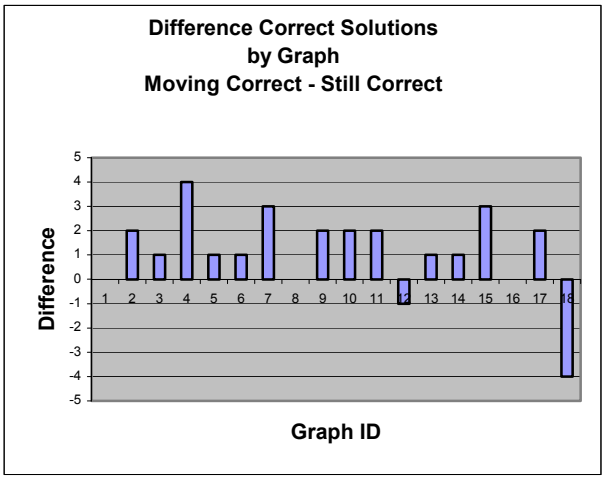

Figure 7

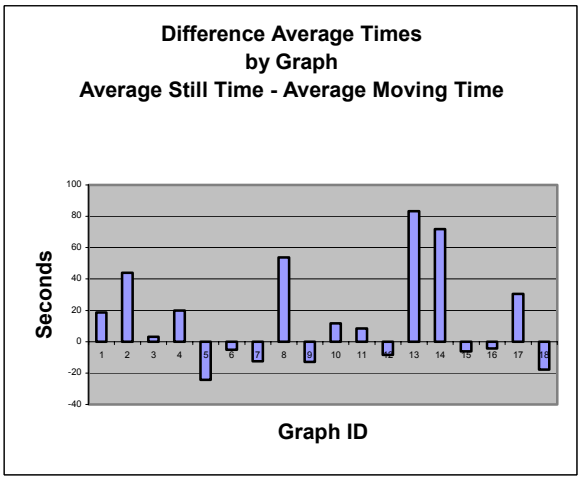

Figure 8

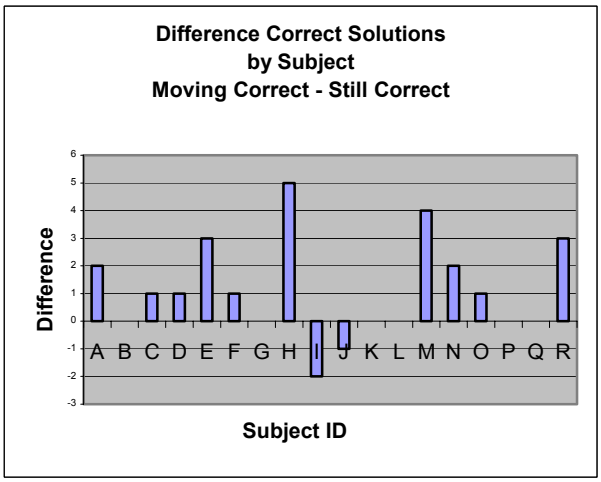

Figure 9

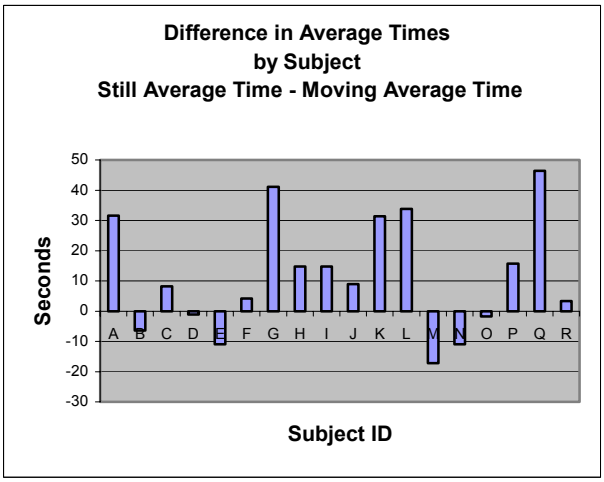

Figure 10

Our interpretation of the results: There are clearly indications that movement helps in solving the set task, but they are not conclusive. For some graphs we see a substantial improvement in subjects' performance particularly in Graphs 13 and 14, see Figures 4 and 5. In Graph 13, there is a high incidence of node-edge occlusion, particularly around those nodes and edges contained within the shortest path, but this is not the case in Graph 14 so some other factor must come in to play. Performance over Graph 5 is slightly better when the graph is moving, but the poorest performance for time for moving graphs overall, is for Graphs 5 and 18, see Figures 3 and 6. However, Graph 18 has some occlusion within the shortest path area; but Graph 5 has no occlusion. Closer inspection of Graphs 5 and 18 reveals a common feature: if the shortest path is considered over an area, in both cases the area covered by the shortest path is small relative to that covered by the graph as a whole. Further, in Graph 14, which the subjects found less difficult when moving, the area covered by the shortest path is relatively large, almost half that of the graph as a whole. The table shown in Figure 11 summarises these observations, with the shaded rows indicating better performance over moving graphs, unshaded better over still graphs and asterisks the presence of the feature in the column heading.

graph	high node/edge occlusion	node/edge occlusion on path	small area	large area
13	*	*		
14		*		*
5			*	
18		*	*	

Figure 11

The relative area of the shortest path may well be a crucial factor with regard to the ease of finding that shortest path.

Finally, it is worth noting that if we consider individual subject performance the degree of subject benefit from movement also varies slightly. There are just two subjects who performed worse on average for accuracy and 6 subjects who performed worse for time when the graphs were moving, however, of those 6 the actual differences in average times were less than 2 seconds. Hence, whilst most subjects saw an improvement in overall performance, there are indications that such movement may not always be helpful for all people.

5: Concluding Remarks

Adding movement to graph diagrams shows promise as a computationally inexpensive way to make the graph structure clearer and easier to understand. Although there are many different ways to make a graph move, we have conducted an empirical trial into the effect of one kind of movement on the performance of one task on a single graph laid out in different random ways. In this trial, the subjects' performance on moving graphs was clearly better overall, although we cannot claim that the difference is statistically significant. However, we plan to do more trials with different kinds of graph layout and different kinds of movement. It does look likely that movement mainly helps with occlusion, and it would be interesting to quantify the level of occlusion in graph diagrams so that we can determine whether resolving occlusion is the only benefit.

If graph movement is going to be useful in real-life graph display software then it will need to be used in conjunction with other tools for making graphs more understandable, such as graph layout algorithms and automatic graph rendering techniques. For example, if our trial graphs had been laid out neatly (as in Figure 2) then adding movement would not have been useful, but most real life graphs are much more complex than this and automatic graph layout algorithms still result in diagrams with occlusion. Also, our graphs are not typical in that the nodes are very small and they have no added detail such as node and edge names that make occlusion more likely.

Another possible role for movement in graph displays could be in situations where the use of automatic layout algorithms is not practical because the graph is too large or because it is changing in real time. It is also clear that if this technique is to be applied in a real world situation then it will be necessary to allow the users control over when, where and how the movement occurs. At its simplest this is likely to be a 'Move' button on graph display software, which switches on and off the movement of graphs when desired. The software should also allow the configuration of the movement, to tune the particular movement method, movement speed and movement distance for the needs of a particular user.

References

1. S. Feiner, D. Salesin and T. Banchoff. Dial: A Diagrammatic Animation Language. IEEE Computer Graphics and Applications 2,9 pp. 43-54. 1982.
2. C. Friedrich and P. Eades. Graph Drawing in Motion. Vol. 6, no. 3, pp. 353-370. 2002.
3. F. Höfting, E. Wanke, A. Balmoŝan and C. Bergmann. 1st Grade - A System for Implementation, Testing and Animation of Graph Algorithms. LNCS 665, 706-7. 1993.
4. H.C. Purchase, R.F. Cohen, and M. James. Validating Graph Drawing Aesthetics. GD95, LNCS 1027, 435-446. 1995.
5. P. J. Rodgers and N. Vidal. Graph Algorithm Animation with Grrr. In Agtive99: Applications of Graph Transformations with Industrial Relevance, LNCS 1779, pages 379-394. 2000.
6. C. Ware, G. Frank. Evaluating Stereo and Motion Cues for Visualizing Information Nets in Three Dimensions. ACM Transactions on Graphics Vol.15, no. 2, pp. 121-140. 1996