MENTAL MAP PRESERVATION PRINCIPLES AND RELATED MEASUREMENTS FOR QUANTITATIVE EVALUATION OF FORCE-DIRECTED GRAPH LAYOUT ALGORITHM BEHAVIOUR

by

David S. Bergman
B.Sc., Simon Fraser University, 1999

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

In the
School
of
Interactive Arts and Technology

SIMON FRASER UNIVERSITY
Spring 2010

Approval

Name: **David Steffen Bergman**

Degree: **Master of Science**

Title of Thesis: **Mental Map Preservation Principles and Related Measurements for Quantitative Evaluation of Force-Directed Graph Layout Algorithm Behaviour**

Examining Committee:

Chair:

Dr. Marek Hatala
Associate Professor and Graduate Chair
School of Interactive Arts and Technology

Dr. John Dill
Senior Supervisor
Professor Emeritus
School of Interactive Arts and Technology

Dr. Thomas Shermer
Supervisor
Professor
Computing Science

Dr. Tom Calvert
Supervisor
Professor Emeritus
School of Interactive Arts and Technology

Dr. Chris Shaw
External Examiner
Associate Professor
School of Interactive Arts and Technology

Date Defended/Approved: April 1st, 2010

Declaration of
Partial Copyright Licence

Abstract

Users working with graph layouts form mental maps of the layout's parts and relative structure. When a graph layout is changed, the user's mental map should be preserved. In the field of graph drawing, there is a need for quantitative and objective measurements to describe and compare layout algorithm behaviour as it pertains to maintenance of the mental map. This thesis presents several mental map preservation principles gathered from the literature, and generates related measurements that can be used to statistically characterize and test for significant differences between layout algorithm behaviour. Two well-known and similar Force-Directed layout algorithms (Kamada-Kawai and Fruchterman-Reingold) are compared, and the results show statistically significant differences. The measurements, statistics, and methodology presented in this thesis may be helpful to layout algorithm designers and graph layout system designers who want to be able to quantitatively and objectively test the algorithms on which they depend.

Keywords: graph; drawing; layout; algorithm; measurement; mental map

Table of Contents

List of Figures

1. Introduction

A graph layout (graph drawing) is a common information visualization tool used to represent a graph's data. Graph layouts communicate complex information by allowing a user to rapidly comprehend relationships and patterns in the nodes and links [55][85][56][87][88][68]. By viewing and manipulating a graph layout, a user develops a "mental map" which represents their ongoing understanding of the relationships between the nodes and links [27]. Some layouts are better than others at clearly conveying information [55]. A graph drawing is "unpleasant," "bad," a "monstrosity," or a "mess" if links and vertices are occluded and if the shape is not evident [36]. As a layout adjustment algorithm changes a layout by manipulating the node positions, the layout is relearned and the user's mental map is updated and thereby preserved [9]. However, if a layout is changed considerably or becomes a tangled mess, the user may not be able to maintain their understanding of the relationships between the graph elements, and the meaning becomes lost [27][60][10]. Therefore, a layout algorithm's behaviour should take into account a user's need to preserve both readability and their mental map. However, there is a need for measurements that can objectively describe how layout algorithms behave [55][9]. These measurements should be based on accepted principles of mental map preservation from the graph drawing community and linked to cognition concepts from psychology (e.g. Gestalt principles).

A layout adjustment algorithm designer and a graph layout application designer can facilitate perception of graph information. However, these designers typically do development in an ad hoc manner [81][36]. This type of development lacks objective empirical evaluation that prevents adequate comparisons of algorithms with respect to the preservation of a user's mental map. The lack of objective algorithm testing also inhibits algorithm designers and information visualization system designers from making constructive design choices [67][44][74].

The ability to compare layout algorithms can help algorithm designers objectively create and incrementally improve their work. A variety of layout algorithms have been developed, [26] but they are often developed without adequate comparison to existing algorithms. In 1995, Himsolt stated that there is no common basis for objective comparison of graph layout algorithms [44],

and several years later, [9] and [67] are still saying there is a need for formalization of algorithm comparison. One class of layout algorithms that could benefit from quantitative and objective comparison is known as Force-Directed (FD). This type of algorithm iteratively adjusts a layout using spring 'forces' to push and pull nodes. Typically, FD layout adjustment algorithms are used to aid user perception of graph layouts (e.g. by reducing node overlap, reducing the number of link crossings). Two well known Force-Directed layout adjustment algorithms are those developed by Kamada and Kawai (KK) [46], and Fruchterman and Reingold (FR) [36]. As a FD algorithm gradually changes a layout, eventually the nodes will come to rest at which point the layout has reached a minimal energy state (a balanced stable state) [28]. As FD algorithms move nodes, the shape and orientation of the layout is changed. Therefore, to avoid accidentally disturbing a user's mental map during the cycles of a layout algorithm's actions, it would be helpful if measurements were available to help to compare the behaviour of layout algorithms with respect to principles of maintenance of a user's mental map.

Though mental map preservation has been mentioned for many years [24], little attention has been given to development and application of measures specific to a layout algorithm's preservation of a mental map [44]. Over the past few decades, studies have examined how a user's understanding of a graph layout can be preserved [27][53][35][65]. A small number of principles relating to mental map preservation are consistently supported by the field of graph layout research [27][44][56][9][53][55][88]. Generally, the principles stem from the fact that a user's perception and understanding of a graph layout is strongly tied to the graph's geometric layout, and so preserving the physical attributes and readability of a layout is important. My research uses these principles as the basis for simple measurements that can be used to evaluate how a layout adjustment algorithm affects a layout. These mental map-related measurements can also be used to objectively evaluate and statistically characterize how layout adjustment algorithms behave relative to one another.

In summary, this thesis presents principles from the graph drawing literature and links them to Gestalt principles from Psychology. The measures that arise can be used to characterize layout algorithm behaviour and examine the algorithm's ability to preserve a hypothetical user's mental map. Chapter 2 reviews topics on graph layout and the mental map that pertain to this study. The quantitative and objective measurements presented in this study are based on principles from the graph drawing literature. The principles and measurements are elaborated in Chapter 3. The measurements were used to statistically characterize the behaviours of two well known Force-

Directed layout adjustment algorithms: KK and FR. The statistics and comparison experiment are described in Chapter 4. The results are discussed in Chapter 5, and conclusions are made in Chapter 6. The ability to objectively measure certain layout algorithm behaviours may allow layout algorithm designers and information visualization application designers to improve their graph layout work as it relates to the preservation of a user's mental map.

2. Literature Review

2.1 Graph Layout Adjustment

A graph layout is a common way to visualize graph information [43]. In a layout, the nodes represent data elements and their relationships are indicated by links that connect the nodes (Figure 2.1.1). Manually drawing a layout of a graph is a time-consuming and detail-intensive chore [60]. Automatic graph layout adjustment, by the use of layout algorithms, releases system users from such chores and can guide them in finding useful information at a minimal cost [60][14][72]. Layout algorithms help users draw graphs that are easy to understand and remember [27].

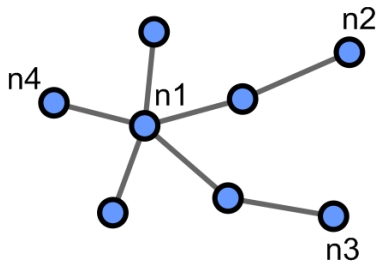
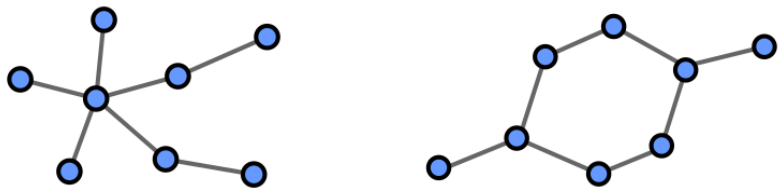


Figure 2.1.1: A graph layout showing relationships between the nodes. The graph has 8 nodes and 7 links. The nodes are represented by circles and links are line segments that join nodes. Although the figure shows nodes as circles occupying area, in this research, nodes are treated as vertices with no size. In this graph drawing, nodes n1 and n4 are directly linked. However, n1, n2, and n3 are indirectly linked.

A graph layout can be created from scratch (e.g. nodes positioned for the first time, "layout creation") or adjusted from its current state (e.g. nodes moved, "layout adjustment"). With a layout creation algorithm, the whole layout can be computed before anything is shown [32],

whereas, a layout adjustment algorithm changes an existing drawing. When a layout is redrawn, the original drawing should be preserved as much as possible to help a user maintain their understanding of the original drawing [9]. Therefore, graph layout creation is not suited for interactive applications where a user may repeatedly alter the graph drawing [9]. When a layout is recreated, the original layout may be completely rearranged, and the rearrangement will destroy the user's mental map of the original drawing [60][9]. If the new layout is much different than the original, the user would have to spend a lot of time relearning the graph drawing [53][9]. Therefore, this research focuses on layout adjustment.

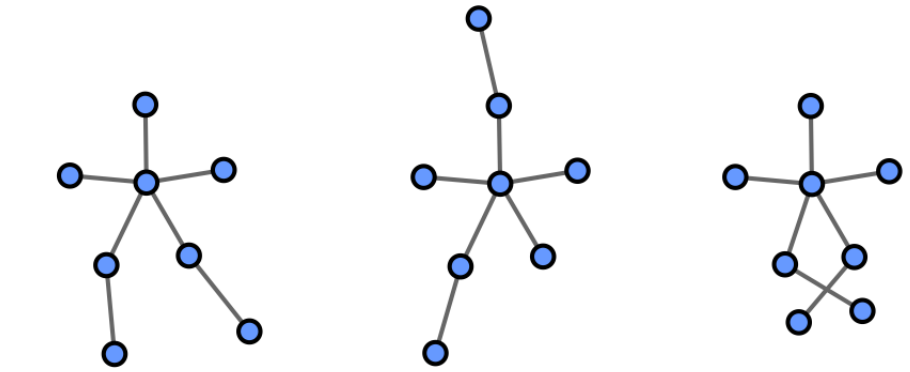
Graph layouts help visualize abstract elements and the nature of the relationships between them (i.e the layout's structure). The information in a graph layout can not only be embedded in the graph elements, but also in the layout itself. So, meaning is embedded in graph layouts based on node positions and the network represented by the pattern of specific linkages between them (Figure 2.1.2) [22][87]. A user's cognitive load is the work required to remember details of a graph layout; and, research suggests that the use of graph layouts helps users reduce cognitive load because visual perception of the layout provides a structural description of the information that takes over some of the cognitive work required by the user [41].



a) b)

Figure 2.1.2: Two different layouts of two different graphs. The layouts show different patterns inherent in the graphs' structures. Each graph drawing has 8 nodes and the link lengths are all approximately equal. Drawing (a) has a link density of 7 links for 8 nodes (7/8 = 0.875) and b) has a link density of 8 links for 8 nodes (8/8 = 1). Figure (a) shows that a high link density at one node can lead to a region of high local node density, whereas in (b), the low link density for each node of the graph leads to lower node density in the center region of the graph drawing. So, although a graph's link density may be higher, there may not be areas of local node density.

A layout adjustment changes node positions and the link attributes (e.g. length, orientation) since each link is dependent on two nodes. Figure 2.1.3 shows three layouts with identical graph data but drastically different layout shapes due to altered node positions. Figure 2.1.4 shows how node position adjustment can affect the link crossing angles of a layout.



a) b) c)

Figure 2.1.3: Three different drawings of the same graph. In layouts (b) and (c), the node positions have been adjusted from (a), but the graphs (the underlying data) are the same and any of the layout versions could lead to the others. The node links are all preserved and the link lengths are all preserved between the versions. In (b), one arm of the layout is on the other side and there are no link crossings. In (c), the arms have been crossed resulting in one link crossing.

a) b)

Figure 2.1.4: Two graph drawings showing different link crossing angles for the same graph. Layout (a) shows a large link crossing angle, whereas (b) shows a small link crossing angle. Maximizing the link crossing angle is one layout criterion that is important for preserving the legibility of a layout [55][88].

2.2 Improving Layouts

Graph layout algorithms try to optimize visual characteristics (layout attributes) to produce useful representations that are more intuitive (easier to perceive and understand) to users [86]. The attributes determine a layout's readability and structure - its usefulness to a user. A few examples of layout attributes that might be optimized by a layout adjustment algorithm include reducing the variability of link lengths, reducing the number of link crossings, or maximizing the link crossing angles (Figure 2.1.4).

The exact definition of a 'good' or 'bad' layout is up to the designer of the layout algorithm, or the designer of the software system that uses the layout algorithm. Designers need to have some idea about which layout attributes they are most concerned with. One way to do this is to use principles that are related to the preservation of the mental map (see Chapter 3) to decide what a good layout algorithm should try to accomplish. Regardless of the relative importance of graph drawing principles that suggest one layout is better than another with respect to certain layout attributes, if layout adjustment algorithms are designed with no consideration for their ultimate purpose, useful visualizations of the graph information may not be produced [70]. Adhering to all the principles equally and at the same time is generally not possible, so a designer will have to choose which criterion is most important depending on the data, the users, and the context [85].

Usually, not all layout criteria can be simultaneously optimized or achieved in one graph layout 'solution' [34][55][33], but precise optimization is not always necessary [44]. Often striving for a mixed balance of partially optimized criteria is better than trying to optimize only one criterion. However, although the importance of graph layout criteria have been examined in user studies [72][70][88], the most important criteria is not known and it is not known exactly how the various criteria should be combined to create better graph drawings [44].

The choice of which criterion is the most important, or which combinations of criteria are the most important to optimize is outside the scope of this research. Deciding which are the most important layout principles and under what circumstances is clearly a very difficult task. For example, Ware, Purchase, Colpoys, and McGill suggest that after the length of a link, the link continuity and number of link crossings are the most important factors [88]. However, Purchase, Carrington, and Allder claim that reducing the number of link crossings is the most important layout criterion to consider [70]. It is commonly upheld that link crossings should be avoided [44], but depending on the graph, certain layout criteria may have to be abandoned because a tradeoff often occurs when different competing layout criteria are considered [53][46][82].

Preserved orthogonal ordering (preserving the relative positions of nodes) is another layout criteria that is thought to be very important by some, but not by all. It is one of the main properties to consider according to [60][82][24] and [53], but is ranked low according to research by [70].

Another tradeoff in layout adjustment is reflected by the fact that proximity relationships among nodes are important to preserve for perception of clustering [82]; however, keeping nodes close, while preserving the nodes' orthogonal ordering, may prevent a potential lowering in link crossings that could occur if the nodes were allowed to slightly migrate.

Additionally, many layout algorithms execute iteratively, producing several intermediate positions for nodes. Therefore, nodes can be thought of as moving along a path from initial to final position. In some cases, it may be beneficial to allow nodes to move along non-optimal paths (i.e. possibly ruining the layout with respect to certain layout criteria) to avoid link crossings, or it might be advantageous to allow for a few link crossings rather than watch the graph untangle in a confusing way [34].

Some researchers believe that, in terms of human understanding of a graph layout, the balance of emphasis on various layout criteria is at least as important as minimizing the number of link crossings [46]. Various graph layout factors will have to be experimentally tested to see what works under which conditions and with what types of data [36][58][2][72]. From a perception research point of view, the area of graph layout research is quite open. Also, because there is some disagreement about the priority of graph layout criteria, the area of research based on perception of graph layouts and the way people use them needs more attention. These topics are outside the scope of this thesis.

In layout conditions in various contexts (different number of nodes, different types of data, etc.), layout criteria may be at odds with one another, but there is no easy way to decide which cognitive principles are most important [91]. The mental map preservation principles described in Chapter 3 can provide some guidance as to how to adjust and improve a graph layout, but I leave the exact definition of what makes a good or bad layout up to the designers. My research makes no claims about the tradeoffs between the benefits or costs of perception and information foraging with respect to the principles described in Chapter 3. I am merely presenting the principles as a consistently upheld set of layout criteria for examining if a layout adjustment algorithm inhibits or facilitates maintenance of the mental map.

A graph layout adjustment can be considered a criteria optimization problem where the goal is for an algorithm to find a layout that maximizes benefits [85]. The layout adjustment algorithm attempts to optimize the benefits of the various criteria to find an acceptable solution. As noted above, optimizing a layout can have a wide range of meanings, but generally the algorithm is trying to improve the layout in one or more ways according to criteria set by the algorithm's designer. Trying to optimize a layout can be seen as an NP-hard problem where many solutions may satisfy the hard requirements of node separation or other soft constraints [34][40][85], but different techniques will be more or less efficient. Although most interesting layout problems are very challenging, feasible solutions that have near-optimal results are sufficient. Therefore, the use of heuristics (e.g. forces) to generate layout solutions is an acceptable and welcome practice [21].

2.3 Force-Directed Layout Algorithms: Kamada-Kawai and Fruchterman-Reingold

This thesis is concerned with looking at how a layout adjustment algorithm changes a layout and how those changes are related to principles based on the preservation of a user's understanding of a graph layout. To investigate the measures and statistics that come from the layout principles I have collected, I decided to use two well known algorithms to perform illustrative testing as a basic proof of concept. The Kamada-Kawai (KK) and Fruchterman-Reingold (FR) algorithms come from a class of layout adjustment algorithms for general graphs called Force-Directed (FD) [46][36].

There has been substantial interest in FD methods because they are simple, applicable to general graphs, and they typically lead to aesthetically pleasing results [37][17]. In FD layout adjustment, the links are treated as though they are springs that connect between the linked nodes. Typically, the nodes are represented as points, and they are acted on by 'forces' that arise due to node positions and link lengths. These metaphoric forces act to displace the nodes, but there is no acceleration as would be seen with real forces.

Force-Directed methods take a nondeterministic approach to layout adjustment that is usually applied to a random initial layout [43]. Graph layout structure arises from initial random clouds of points [39]. Depending on the specific adjustment algorithm used, the forces may change a layout in various ways. For example, repulsions between nodes can help evenly distribute nodes in the drawing space [85]. However, the link lengths can also play a role as metaphoric springs that repel or attract connected nodes based on a link's length. Typically, all links in a node-link system are assigned an 'optimal' desired link length (by some calculation). If a link is shorter than the desired resting length then the nodes at either end are repulsed away from one another. But, if the link is longer than the resting link length, the nodes are drawn together. The balance of repulsions and attractions allows a graph layout to arrive at a resting state where the metaphoric energy is minimized, and the layout becomes stable.

When the nodes are moved, the graph layout is changed, and an initial layout (L) becomes a new layout (L′). Although the layout has changed, the underlying graph data and linkages between the nodes remains the same. Preservation of a layout with respect to various layout attributes plays an important role in the preservation of a user's mental map.

2.4 The Mental Map

The mental map concept was first proposed by Eades in 1991 [53]. A user's mental map is a persistent contextual awareness and understanding of graph layout information that is created and exists in a user's mind. The graph drawing literature describes it not as an image, but as an ongoing understanding of how the graph is structured. A user's mental map may include the shape of the drawn graph and landmarks (e.g. large nodes, node clusters, white space) to facilitate navigation, problem solving, and persistent understanding of the information [27]. This understanding is learned and should be maintained as the layout changes [27].

The mental map described in this thesis is based on descriptions from the graph drawing literature. While the field of Psychology has described similar ideas (e.g. cognitive maps), my research is concerned with what the graph drawing community has to say about this topic. Certainly graph drawing studies that consider the mental map can be aided by Psychology research, but this study uses graph drawing ideas about shape and patterns and so restricts itself to the meaning of mental map and mental map preservation that have been developed in the graph layout literature.

2.4.1 The Mental Map is Formed and Maintained

The mental map is formed when a user sees, interacts with, and learns about a graph layout's structure [53]. Just as a person can form an understanding of a geographical map, when a user interacts with a graph layout, they form an understanding of the relative positions and relationships that are created by the links and nodes. Not only are the elements important, but the shape and pattern of their related geometric positions and linkages between the elements are also crucial for an understanding of the information represented in a layout [25][60].

The mental map gets updated and relearned as adjustments are made to the layout. A user's current mental map may be quite different than their initial understanding of the graph. In the context of graph layout adjustment methods, preserving a user's mental map means attempting to ensure the graph layout algorithm does not do anything to harm their understanding of the graph

and its relational constituents. When a layout is altered, preserving the mental map is crucial for effective use of a system by a user [27][32][20][34] because this allows the user to quickly recognize the graph's new layout [53].

2.4.2 Layout Changes may be Detrimental to Preservation of the Mental Map

A layout's configuration should not be changed considerably since this can disturb or destroy a user's mental map [47][60][82][53][20][27][20]. Changes to the graph drawing usually ruin the layout in some local manner [27] but the mental map may not be destroyed. For example, the addition of a link may cross existing links and make the layout less readable [27]. Local changes only affect one part of the layout, and if the layout changes are small, a user's understanding can be preserved. However, if the layout changes are large (e.g. via large global changes or many small local changes), then the final graph drawing is more different. Therefore, a user's mental map may not be preserved if their original understanding of the initial layout is much different than their current understanding of the new graph layout. For example, if a layout is completely re-arranged (or re-created), the user's mental map is destroyed [27].

The graph drawing literature supports the idea that mental map preservation is related to the degree of change from one layout to another (see Chapter 3) [32]. According to the graph drawing literature, the degree of change to a layout is inversely proportional to the maintenance of a user's mental map [32]. Figure 2.4.2.1 demonstrates that when a new layout differs substantially from its previous drawing, a user may have to spend considerable time relearning the new layout [53].

FD layout adjustment algorithms iteratively change a layout such that the algorithm steps are drawn as the changes occur. An important source of visual information (in a number of application fields) [15] comes from motion [1]. When nodes in a graph layout move from an initial layout position to a final layout position, node movement is perceived. In the field of computer vision, the primary focus has been the displacement of points between frames over time [1][3] where two images are required to extract motion information [83].

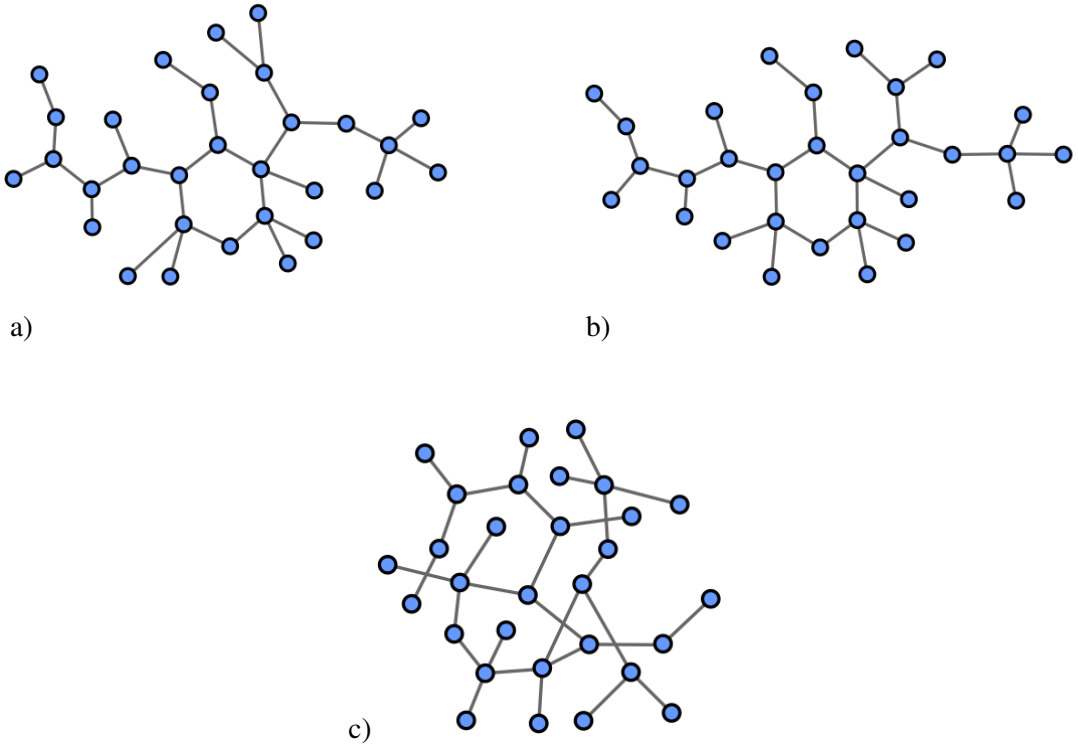


Figure 2.4.2.1: One graph of 30 nodes in three different conformations. Figure (a) is an initial layout, a slightly adjusted version is shown in (b), and (c) is a newly created layout. The first layout is an example of what a user might have been working with. The second layout is an example of what a Force-Directed adjustment algorithm might have done to (a). Layout (c) is an example of what the graph might look like if it were laid out from scratch. The relationship between (a) and (b) is clearly evident, but because (c) is quite different in shape, it is more challenging to see how (c) relates to either (a) or (b) (or even verify that they are the same graph). A user's mental map is more likely to be preserved between (a) and (b) (versus (a) and (c)). Principles related to the preservation of a user's mental map with respect to layout changes are outlined and described in Chapter 3.

In the field of graph layout, precise motion information is obtained by calculating changes from L to L′. The motion information derived from the calculations can be used to create animation frames by interpolation between the stepwise changes to a layout [34]. This animation can help visualize how a layout algorithm has had an effect on the initial layout. The motion information could be used to inform a layout designer that the amount of motion, due to the layout algorithm

used, should be reduced to help preserve a user's mental map [32][53][24][34][9]. Alternately, the designer may wish to create an algorithm that produces small/smooth changes [16] by increasing the frame rate of the layout animation (i.e. by increasing the algorithm cycles per second) [34].

2.4.3 The Mental Map Concept is Related to Layout Principles and Cognitive Theory

Data visualizations have the goal of optimizing the communication of information through improved representation [4]. Even though some research has shown a particular layout criterion to be the most important (e.g. [71]), in the context of a particular visualization system, it is difficult to conclude exactly which layout attributes should be optimized [4]. If researchers can formally determine the optimal goal for quality graph visualizations, and if the optimization can be connected to cognitive theory, then future research can make good progress in developing and evaluating tools for visualizing layout quality [4].

It is difficult to connect graph layout adjustment to cognitive theory, but there are supported mental map preservation principles in the graph layout literature that can be connected to certain Gestalt principles from Psychology. Also, those layout principles can be used to derive measurements that can characterize layout algorithm behaviours with respect to preservation of the mental map. In Chapter 3, key layout principles, which are supported in the graph drawing literature, are described and used to develop measurements. These measurements can be used to objectively characterize how a layout algorithm behaves with respect to specific layout attributes related to the principles. The measurement results can be used to draw conclusions about if an algorithm helps to preserve a hypothetical user's mental map.

3. Mental Map Preservation Principles and Related Measurements

Force-Directed (FD) layout algorithms were first proposed by Eades [28]. In this type of algorithm, the nodes are connected by springs and the nodes are acted on by forces until these springs reach an equilibrium length. Depending on the specific FD algorithm, the forces that push and pull the nodes may act between all nodes or just directly-linked nodes. Force-Directed algorithms, such as KK and FR, are based on Eades' Spring Embedder algorithm [44]. FD algorithms are well suited for incremental layout since they can draw the layout after each algorithmic iteration (cycle) [32]. Each cycle, the node positions are recalculated based on the sum of forces the nodes experience (attraction, repulsion, and others specific to the FD algorithm) and the nodes are drawn at their new locations. The successive layout drawings, outputted to the screen over time, form an animation of the incremental layout changes like the frames of a movie. Just as in movies, it is possible to create intermediate frames to make the animation appear more continuous, smooth, and therefore effective [32][5]. Because the nodes are seen to move through time in the frames of the layout algorithm iteration steps (as in phi motion) [90], it is possible for a user to visually track how nodes change position in the iteratively changing layout. Different layout algorithms will produce different amounts of change between algorithm iterations [32].

Force-Directed algorithms act by energy reduction. These algorithms are based on an optimization problem, and when a layout algorithm has found a solution, the layout is said to be stable because the energy in the system is at a minimum [32][28][46][36][33][18][43] The forces (pseudo-forces) in the system are related to the energy, and these forces affect how far nodes want to move. The energy functions are different for different layout algorithms, but energy is related to the amount of node movement. Therefore, when there is little energy in a layout, the nodes are not moving much or at all (static equilibria) each algorithm cycle [36]. Generally speaking, the number of iterations required to find a layout solution is poorly understood [85]. Typically, more cycles are needed for larger graphs [32], and therefore, the animation will take longer. Note that as a layout changes from L to L' (initial to final layout), it is possible for a node to move considerably during the algorithmic cycles, but the node may end up at a position that is not far from its initial position (large activity but small amount of node displacement).

3.1 Layout Adjustment Principles for Mental Map Preservation

Users of graph visualization systems typically spend a considerable amount of time getting familiar with the layout [27]. Therefore, changes need to be made carefully. Each adjusted layout should conform to specific criteria to improve the meaning and readability of the drawings [8]. Researchers have suggested several graph drawing characteristics that should be taken into account, when considering a layout's change from an initial to final layout, in order to maintain a user's mental map.

The general shape of a graph layout is created by the position of the nodes and the links that join them. A user may edit a graph's layout (e.g. by moving nodes) and then they may want a layout adjustment algorithm to help find another layout (e.g. to lower the number of link crossings between nodes). Preservation of a user's mental map is the ability to maintain an understanding of the graph's structure over time as the layout changes [27]. If a layout adjustment algorithm acts on a layout, the user's mental map may be affected [32]. Maintaining clustering and the general shape of a graph will help maintain a user's mental map [27][60][56]. Understanding these principles may help a designer decide how to create a layout algorithm that tries to avoid potential problems that may disrupt or destroy the user's mental map.

If a graph drawing changes considerably, a user has to re-orient himself [47] by relearning where the parts of the graph drawing have moved to and relearning/reaffirming the relationships of the parts (i.e. where did nodes move and where are the links that connect them?). To help a user maintain their mental map, less change from L to L′ is preferable, or else the mental map may be damaged or destroyed [27][60][82][53][20][32][47][34][56]. Preservation of the mental map can be facilitated by reducing the magnitude of node movements from cycle to cycle so the changes are smooth (as opposed to large discrete changes) [88][34][32][61][13].

The idea that smooth changes to a visualization are important to the persistence of understanding is also found in the field of video compression research. Parts of a drawing can be perceived and understood to have meaning that can be maintained as the drawing changes over time (e.g. optical flow) [84][5]. A layout algorithm iteratively produces layouts (from L to L′) that are viewed like animation frames. These frames of a changing graph layout are separate images, and the mental

map relies on a person's ability to recognize the apparent motion of nodes. The human visual system is very sensitive to motion [68], and reducing the apparent changes to a layout is similar to the goals of motion compensation in video compression research [11]. The fields of graph layout and video compression both attempt to preserve a user's perception of what is in a visual scene, from one frame to the next [27][32][84][11][52].

A user's ability to recognize apparent motion where there is actually none is referred to as the phi phenomenon (also known as stroboscopic movement perception), and its roots are in Gestalt Psychology [90][80]. When a layout is drawn, small adjustments to node positions are visualized with nodes at slightly different positions, and the user perceives the changes over time (from frame to frame) as motion. Large changes may mean the user cannot perceive continuity of movement [49] since a node may have moved far from where it was last seen. This concept coincides with the Gestalt principle of contiguity (elements that are spatially and temporally proximal allow the mind to perceive movement) [61][54][13]. It is important to consider these concepts of apparent motion when studying how algorithmic behaviour can preserve the mental map as a layout changes from L to L′.

Eight layout principles were identified in the graph drawing literature that, if followed, may tend to help a user maintain their mental map of a changing layout. The principles, described in detail below, are:

1. minimize time to layout completion
2. maximize number of cycles per second
3. minimize node movements
4. preserve linked-node clustering
5. preserve orthogonal ordering (relative node positions)
6. minimize link lengths
7. minimize number of link crossings
8. maximize link crossing angles

These principles stem from the idea that a layout's structure should remain readable and recognizable as it changes from L to L′ in order to help a user maintain their mental map [27][60][53][32]. This means that although a layout changes, a user's understanding of the graph is maintained by relearning and thereby reaffirming their ongoing understanding of the layout.

Moreover, layout changes should be minimized (principles 1-5) while readability attributes/factors should be maintained or enhanced (principles 6-8). It has been suggested that making small changes to fine tune a layout may improve readability [66]. Readability is important in the preservation of a user's mental map [53].

Below are more detailed descriptions of the graph drawing principles that are related to the maintenance of a user's mental map.

3.1.1 Principle 1: Minimize Time to Layout Completion

The amount of time for an algorithm to adjust a layout to a completed state (i.e. the energy in the layout is minimized sufficiently and the layout algorithm is finished) is a measure related to the preservation of the mental map. This concept is supported by the Gestalt principle of contiguity (elements that are spatially and temporally proximal allows the mind to perceive movement) [61][54][13]. For example, if an algorithm takes a small amount of time to complete, the user does not have to pay attention for long, and the temporal collection of images (from each algorithm iteration) is smaller. However, if the time to completion from L to L′ is larger, it may be more difficult for a user to perceive how the node positions have all changed over the longer time period.

3.1.2 Principle 2: Maximize Smoothness of Geometric Change

Mental map preservation can be assisted by creating a smooth transition from one drawing to the next [34]. It is easier to track a node that moves continuously and smoothly in both space and time as compared to one that jumps from place to place over the cycles of layout adjustment from L to L′ [34]. This spatio-temporal smoothness is affected by a number of factors, including number of cycles, the time between cycles, the total path length between L and L′, the length of individual movements from cycle to cycle, the angle between successive vectors of the motion (related to path curvature) and total net change in position. This notion is supported by the Gestalt principle of contiguity (elements that are spatially and temporally proximal allows the mind to more easily perceive movement) [61][54][13]. For example, a higher frame rate helps a

user perceive moving nodes rather than jumping from position to position. Frame rate is related to the number of drawings per second, and is therefore related to the number of algorithm iterations per second (cycles per second).

3.1.3 Principle 3: Minimize Node Movements

The distance nodes move from L to L′ is important with regards to a user's mental map and their ability to perceive where the nodes have moved to. Since less change between layouts is better for preserving understanding, it clearly follows that less node movement is preferable for maintenance of the mental map [34][56]. It also follows that when there is no change, the user's mental map is preserved [20].

Node positions are related to layout shape, reducing node movements will preserve a layout's shape [9], and preserving the shape is important for maintaining a user's mental map [27][66]. This idea of preserving node positions is related to layout structure and geometric node clustering, but is not concerned with node linkages. Nodes that are close together can be perceived as groups which represent key layout features [9]. This notion of clustering is related to the Gestalt principle of invariance (geometric objects can be recognized independent of scale, translation, or rotation) and the Gestalt principle of proximity (elements that are closer together tend to be perceived together as a grouping) [61][54][13][76]. Because the layout changes happen over time (frame by frame), minimizing node movements is related to the Gestalt principle of contiguity (elements that are spatially and temporally close to one another may be perceived as the same element), and contiguity is related to the Gestalt concept of the phi phenomena.

3.1.4 Principle 4: Preserve Linked-Node Clustering

As with the previous version of clustering, proximity relations of nodes play an important role in the perception of a graph layout and a user's mental map [60][56]. However, this version of clustering is slightly more specific to include only those nodes that are directly linked. Linked-

Node Clusters (i.e. nodes that are directly-linked to one another) represent related objects in a graph layout (e.g. for manipulating clustered nodes based on hierarchy) [31].

3.1.5 Principle 5: Preserve Orthogonal Ordering

Orthogonal ordering is a heuristic that considers the relative ordering of nodes horizontally and vertically. Basically, a change in orthogonal order is registered when one or more of the nodes around a given node are moved to another geometric quadrant (see Figure 3.1.5.1). When node positions are preserved relative to one another (e.g. scaling), the orthogonal ordering is preserved [27].

When orthogonal ordering is measured, it is done for all nodes in a layout. In Figure 3.1.5.1, node n2 happens to start in quadrant 1. If n2 were to move left across the vertical axis into quadrant 2 or down across the horizontal axis into quadrant 4, that would be an orthogonal ordering change of 1 for node n1. If n2 moves both into quadrant 3 (the path is not important as long as it ends in quadrant 3), that move represents a change of 2 because n2 has crossed both the vertical and horizontal axes. If n2 eventually comes to rest in the same quadrant it started in, there is no orthogonal ordering change for n1 due to node n2. Although this example is looking at the ordering around node n1, the same calculation will be done for all the other nodes to get the orthogonal ordering change for the entire layout. To calculate the change in orthogonal ordering for the layout, the change in orthogonal ordering is calculated for all nodes in the layout from L to L′. The total change in orthogonal ordering represents one way to measure the amount of shape change for the graph drawing.

Orthogonal ordering can be examined to see if the shape of the original graph layout is preserved relative to the final layout. Preservation of the general shape of a graph drawing can be tested by examining whether the orthogonal ordering of the nodes in a layout mostly remains the same [27][60][56][24][20]. A layout's shape is related to its parts, and this is related to the Gestalt principles of reification (the parts of a drawing help perception of the whole) and invariance (geometric objects can be recognized independent of scale, translation, or rotation) [54]. It has been suggested that measuring orthogonal ordering differences (L→L′) is very important for examining algorithm behaviour with respect to how layouts are changed [9]. Therefore,

including a measure of preserved orthogonal ordering is important because preservation of the mental map is related to preservation of layout shape.

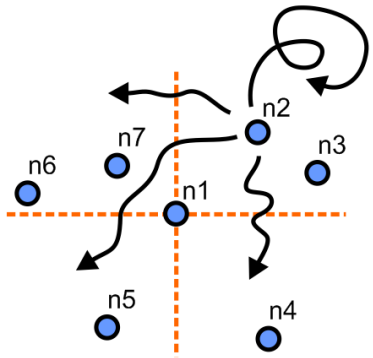


Figure 3.1.5.1: Relative node movement from one quadrant to another around a node represents a change in orthogonal ordering. In this example, the orthogonal ordering changes for node n1 are examined when the only node to move is n2. The arrows represent possible movements for node n2. The arrows are squiggly to represent the idea that the exact path of node n2 is not important as long as it ends up in the geometric quadrant indicated by the head of each arrow. The horizontal and vertical axes that go through node n1 are represented by dotted lines. Note that the typical measurement of orthogonal ordering does not consider links so they are omitted from the figure for clarity.

Note that orthogonal ordering is not preserved when a layout is rotated by a large amount that would cause the relative node positions to change (e.g. imagine all the nodes around n1 rotating by 90 degrees). In this case, even though the rotated shape of the drawing has not changed (it is homeomorphic with the original layout's structure), the shape relative to the viewer has changed in terms of orthogonal ordering [19]. Preserving user orientation is critical [32]. A change in orthogonal order may negatively affect a user's ability to maintain their mental map because the change is an indication that, from user's perspective, the shape is not the same [27]. Although the Gestalt principle of invariance helps a user maintain their mental map by allowing them to still recognize the rotated shape, invariance may not be able to help the user maintain their mental map when the rotation amount is considerable.

3.1.6 Principle 6: Minimize Link Lengths

It has been suggested that long links are hard to follow [47], so it may be that a designer will want to use an algorithm that has the smallest average resulting lengths (minimizing total link lengths) [55]. Uniformity of link lengths implies total balance and this may lead to a better understood graph [46]; therefore, uniform and short link lengths are desirable for preservation of the mental map.

3.1.7 Principle 7: Minimize Number of Link Crossings

There have been suggestions as to what a better layout means in terms of readability. It has been suggested that a less cluttered layout is more readable and better at conveying information effectively [55][30][44][60]. In the context of graph drawing, a visually cluttered layout means that the drawing is hard to perceive (and therefore hard to understand) because the nodes and links form an unnecessarily complicated and visually confusing layout. Many unneeded link crossings will make it hard to see which nodes are connected to each other; therefore, adjusting node positions that reduce the number of link crossings reduces visual clutter [17] improving the readability and the user's comprehension of a layout [21]. Furthermore, measuring the number of link crossings is a common way to judge the readability of a graph drawing [30]. Reducing link crossings to facilitate readability is related to the Gestalt principle of continuity (the mind continues perceiving visual patterns like paths formed by line segments (i.e. the links between nodes)) [87][88]. Where links cross, they may cause mutual visual interference such that any pattern found by continuity may be obscured by the visual clutter. Therefore, to reduce visual clutter and make the layout easier to explore, reducing the number of link crossings is desirable [62][17][6][87][71].

3.1.8 Principle 8: Maximize Link Crossing Angle

The link crossing angle is the smallest angle formed by two intersecting straight links [55]. Maximized link crossing angles are important to facilitate legibility and perception of a graph layout [55][60]. This is supported by neurophysiology research that suggests that link crossings are easier to perceive when they are closer to 90 degrees rather than at sharp angles (e.g. less than 30 degrees) [88]. The idea that the crossing angle should be maximized to facilitate perception is related to the Gestalt principle of continuity (the mind perceives two line segments on either side of an intersecting line as one continuous line) [88].

3.2 Measurements

The measurements presented here are based on the principles described above for the preservation of a user's mental map. Each of the measurements arises from the principles mentioned in the previous section. The measurements are: time to completion, number of cycles, node displacement, node activity, Delta Directly-Linked-Orthogonal Ordering (DDLOO), link length, number of link crossings, and link crossing angle. In the experiment (described in Chapter 4), these measurements are used to generate statistics that allow for characterization and comparison of the two algorithms tested: Kamada-Kawai (KK) and Fruchterman-Reingold (FR).

Each initial layout (L) is treated with KK. A copy of the same initial layout is also treated with FR. The procedure results in a pair of final layouts (L'^{KK} and L'^{FR}) that effectively arose from the same initial layout. Many trial runs are conducted by creating more pairs of initial layouts which result in more pairs of final layouts. The set of initial layouts (S(L)) is the same for both algorithms. The differences between the initial and final layouts can be measured using the measurements described below, and the averages and standard deviations characterize how the algorithms behave (from L to L′). This paired experimental design allows for a direct comparison between the algorithms' general behaviours. The paired runs allow for 2-tailed t-tests to check for similarity between the resulting pairs of statistical data (i.e. $L \rightarrow L'^{KK}$ and $L \rightarrow L'^{FR}$).

3.2.1 Time to Completion

- *This measurement is related to the preservation of a user's mental map by principles 1 and 2: Minimize Time to Layout Completion, and Maximize Smoothness of Geometric Change*

This measurement is the time for an algorithm to run to completion from L to L′ as measured in seconds.

3.2.2 Number of Algorithm Cycles to Completion

- *This measurement is related to the preservation of a user's mental map by principle 2: Maximize Smoothness of Geometric Change*

This measurement represents the number of iterations an algorithm takes to change L to L′. Measuring the number of algorithm cycles and the time to completion are necessary to calculate the number of cycles per second.

3.2.3 Node Displacement

- *This measurement is related to the preservation of a user's mental map by principles 2 and 3: Maximize Smoothness of Geometric Change, and Minimize Node Movements*

Node displacement (for a single node) is the Euclidean distance a point has moved from initial layout to final layout (L→L′) [56][9]. Although a node may experience chaotic oscillating movements in the algorithm iterations between L and L′, the node's displacement may still be very small.

3.2.4 Node Activity

- *This measurement is related to the preservation of a user's mental map by principles 2 and 3: Maximize Smoothness of Geometric Change, and Minimize Node Movements*

The inspiration for this measurement comes from the Sum of Absolute Differences (SAD) metric which is simple, easily calculated, and common in various fields of research including video compression research [5]. SAD is a measure of the total geometric change - regardless of direction. It is not a measure of displacement, but a measure of the amplitude of total directionless movement, i.e. path length. In the field of information visualization and perception, SAD is a widely used metric for quality in block-matching for motion estimation and compensation for video compression and analysis [83].

Node Activity is the sum of absolute values for a node's movements (total travel distance, path length) through all steps as a layout changes from L to L′. The need for this measurement arises because node displacement does not account for how the nodes move as a layout iteratively becomes L′. It may be that, for a particular algorithm, the average node displacement is small, but the total node movement is quite large. For example, the node displacement measure helps us understand that from L to L′, a node has gone from point A to B. But, the measurement does not help us understand how much movement was involved in getting from A to B. It may be that the travel direction changed chaotically, or it may be that the movement of the node overshot B and then came back. This measure cannot tell the difference between these two examples, but it does give an idea of how efficient a node's movement is as compared to the node displacement measurement. Measuring the node activity for a layout further helps explain how a layout algorithm behaves with respect to node movements.

3.2.5 Delta Directly-Linked Orthogonal Ordering (DDLOO)

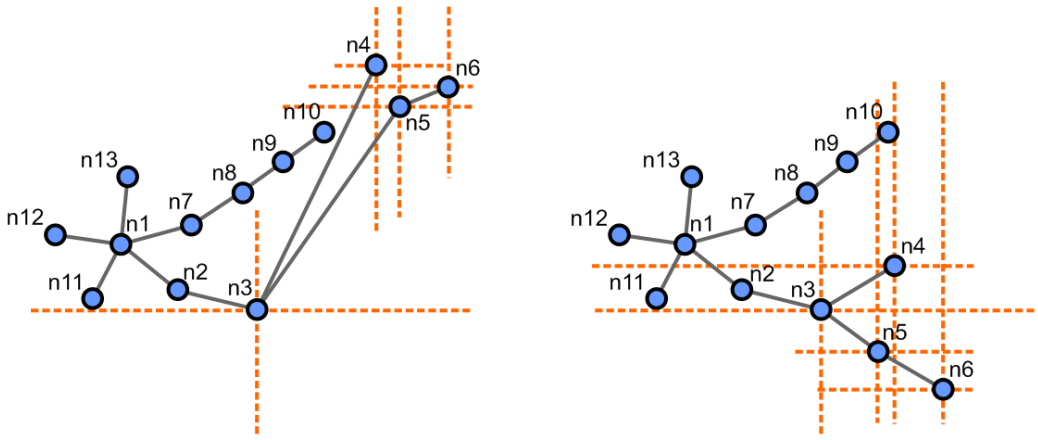
- *This measurement is related to the preservation of a user's mental map by principles 4 and 5: Preserve Linked-Node Clustering, and Preserve Orthogonal Ordering*

Changes to orthogonal ordering in a layout can help describe if the graph layout's shape changes [27][9]. In terms of the macro-structure of a layout, I speculate that examining the way linked neighbors move (local movement) is more interesting and meaningful than looking at the relations between distant indirectly-linked nodes. Delta Directly-Linked Orthogonal Ordering (DDLOO) is a novel method to measure relative node position changes which considers only directly-linked nodes; therefore, the DDLOO measurement helps describe how the shape of a layout is locally changed in terms of the orthogonal ordering of a node's directly-linked neighbors. DDLOO focuses on examining local change because it is relevant to a user's persistent perception of a drawing [19], and therefore, mental map preservation.

The links in a layout help define its structure. As a layout algorithm runs, the links guide the movements and positions of larger graph features. This effect is seen when a long twisted length of linked nodes is allowed to uncoil and straighten itself out. In this example, each part of the shape change is directly related to the neighboring links and nodes. This emphasis on the importance of local structures in graph layout is present in Freire's work and in Watts' "small world" idea in which clusters do not interact strongly with distant nodes, but they communicate strongly with their local area [89][32].

Since links dictate layout structure, logically it follows that DDLOO values should give a more specific measure of how a layout changes shape as a whole. A large DDLOO score for a node indicates that there was more local shape change in that part of the layout. A large average DDLOO score for a layout suggests there were several large local changes. Conversely, measuring the Global Change in Orthogonal Ordering (GCOO) for node positions (regardless of linkages) is more prone to confusing data from nodes that are distantly related. This notion is demonstrated in Figure 3.2.5.1.

Figure 3.2.5.1 and Appendix 1 demonstrate the difference between measuring global and local orthogonal ordering changes (GCOO versus DDLOO). The specifics of the calculations are given in Appendix 1. The general procedure for calculating GCOO includes calculating the node position changes horizontally and vertically. The changes in each dimension are treated separately, and the resulting values are then summed to yield the GCOO value for a layout. The general procedure for calculating DDLOO is exactly the same except that only node position changes are scored between nodes that are directly-linked.



a) b)

Figure 3.2.5.1: Comparing global and local orthogonal ordering changes. Figure (a) is an initial layout, and (b) is a final layout that might arise after applying a layout algorithm (e.g. KK or FR). The final drawing shows a few small adjustments to the overall shape, but the shape is still quite similar to that of L. For this layout change, DDLOO is 4 and GCOO is 49. For illustration purposes, only nodes 4-6 are moved. Vertical and horizontal dotted lines are drawn to show the relative position of a few of the key nodes.

Figure 3.2.5.1 illustrates a simple example of the benefits of DDLOO over GCOO in terms of the ability to specifically measure local relative changes around a node. For example, the movement of n4 has an effect on the overall shape; however, this effect has nothing to do with its relative position to distantly-related nodes (such as n10). Therefore, n4's measure of orthogonal ordering (as a measure of layout shape) should not consider the relative positional change of n10. Another

example is n4 compared to n5, where their relative positions are relevant to n3, but n4 and n5 do not need to know about one another in terms of the general shape because the local shape spawns from n3. Note that if the example were substantially more complex (e.g. twice or three times as many nodes and links), the GCOO score may be drastically affected in chaotic ways when only a few nodes move.

3.2.6 Link Length

- *This measurement is related to the preservation of a user's mental map by principle 6: Minimize Link Lengths*

When a node is repositioned, its links may become shorter or longer. This measure is the distance between connected nodes [53].

3.2.7 Number of Link Crossings

- *This measurement is related to the preservation of a user's mental map by principle 7: Minimize Number of Link Crossings*

The number of link crossings is a count of the total number of link crossings that occur in a layout. A single crossing is counted whenever two links intersect. This value will help determine if an algorithm is good at reducing the number of link crossings [33][44][7][71][88].

3.2.8 Link Crossing Angle

- *This measurement is related to the preservation of a user's mental map by principle 8: Maximize Link Crossing Angle*

The link crossing angle is the minimum of the two angles formed when two links intersect. This measurement will help determine if an algorithm is good at aiding the perception of visual continuity of links that are crossing [88][55].

4. Experimental Setup

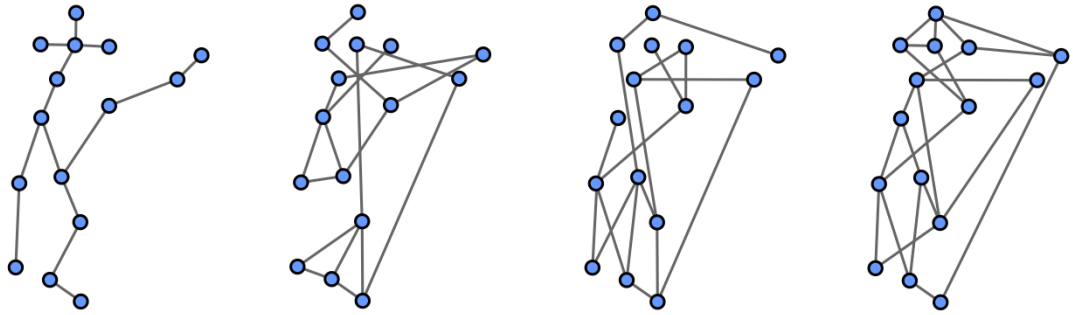
4.1 The Experiment

This experiment was designed to test if the measurements related to preservation of the mental map (described in section 3.2) could be used to find significant differences in certain aspects of graph layout algorithm behaviours.

The experiment began with the creation of a random initial layout (L) of a certain graph size. The experiment uses two graph sizes (15 or 30 nodes). Production of an initial layout is accomplished by randomly choosing (from a uniform distribution of random numbers) a link density threshold (from 1.2 to 1.6 links per node in the layout), and assigning links between random nodes until the graph is completely connected and the randomly determined link density threshold is reached. Random nodes are chosen by choosing two pseudo random integers (each from a uniform distribution of random numbers) between 0 and the number of nodes (not inclusive on the top end). Attaining a random value in this way was accomplished by generating a random number between 0 and 1, multiplying by the number of nodes, and then rounding down to the nearest integer. The index range (0 to number of nodes -1) represents the range of valid indices for all the nodes in the layout. If the two indices chosen are the same, the random values are re-chosen until they are different (because a node cannot have a link to itself). As links are randomly connected between nodes, to prevent too much density around a single node, a maximum number of links of four is imposed. After the links have been assigned, all the nodes are randomly positioned in the layout area (1000x1000 pixel area). The nodes are randomly positioned by choosing two pseudo random integers (each from a uniform distribution of random numbers) between 0 and 1000 (not inclusive on the top end, same method described above) for each node. This range (0 to 999) represents the range of valid x and y values for a node (the drawing area is 1000x1000). The two integer values are assigned to the node's x and y positions, and if the node occupies the same position as another, it is randomly repositioned. This random repositioning of the node continues until it occupies an empty position. This process of choosing and assigning random x and y values continues until all the nodes are positioned in the drawing area. The link

density range and the maximum number of links on a single node are both consistent with the figures in the KK and FR source papers [46][36]. A few valid and invalid initial layouts are presented in Figure 4.1.1.

In Figure 4.1.1, drawing (a) is not a valid initial layout because, although the layout is connected (all nodes are at least indirectly connected to one another by links) and no node has more than 4 links, the link density is only 0.93 (14 links/15 nodes). This link density value does not fall in the valid range (1.2 to 1.6) for this experiment. Drawing (b) has 18 links (link density = 1.2) and is within the valid link density range, and none of the nodes have more than 4 links, but the graph is not connected. Therefore, drawings (a) and (b) could not be valid initial layouts for use in this experiment. Drawing (c) would be valid if a trial's chosen random link density was 1.2 (18 links/15 nodes). Drawing (d) would be valid for a trial with link density of 1.5 (23 links/15 nodes).



a) b) c) d)

Figure 4.1.1: Four possible initial random layouts for graph size 15. Drawings (a) to (d) show various possible initial layouts which may or may not be valid for use in the experiment. The drawings show how a 15 node layout would be different depending on how links are randomly assigned.

Two graph sizes were used in the experiment (15 or 30 nodes per layout). After a valid initial layout (L) was created at a particular graph size (e.g. 15 nodes), a particular layout algorithm was applied iteratively until the changing layout reached a stable final state (L′). At this point, the final layout had a negligible amount of node movement (the energy in the layout had been

sufficiently reduced). Initial and final layouts for a particular algorithm were used to compute the measurements. This process was repeated for the other algorithm using the same initial layout (L). This procedure defines one trial run.

For each graph size, 502 trials were run. Before an algorithm was applied to the set of initial layouts (S(L)), they were copied twice. The first copy was treated with the KK algorithm and this produced a set of final layouts (S(L'^{KK})). The second copy was treated with the FR algorithm and resulted in another set of final layouts (S(L'^{FR})). The original set (S(L)) was preserved for later comparisons to both final layouts (S(L'^{KK}) and S(L'^{FR})). The same number of trials was used for graphs of 15 and 30 nodes which produced S(L'^{KK}_{15}), S(L'^{FR}_{15}), S(L'^{KK}_{30}), and S(L'^{FR}_{30}). The four sets of final layout data are treated separately with regard to measurements and statistics calculations (averages and standard deviations) on the measurements.

The Kamada-Kawai [46] and Fruchterman-Reingold [36] Force-Directed layout algorithms were implemented in C++. Both algorithms function iteratively to gradually change a layout. Each cycle (iteration), the net forces on all nodes are calculated. After each cycle, the current layout is drawn to the screen. The process continues until the layout reaches a stopping condition. The experiment was run on a Pentium 4 (3GHz) in Vista with 1GB DDR400 RAM.

4.1.1 Common Steps to Both KK and FR Algorithms

Both KK and FR algorithms run iteratively, and they perform common steps each time they are run. The time is recorded before the layout starts being adjusted. After an algorithm completes a layout, the time is again recorded. The change in time is calculated, and the number of cycles is recorded.

To allow each algorithm to come to a similar state of completion, the same stopping condition was used for both KK and FR. A layout is considered done if the total node movement for all nodes in a layout (the sum of all individual node movements in an algorithm cycle) is less than two pixels.

Forces in KK and FR are treated as though they are direct adjustments to node displacements. They are not forces in the physics sense which would allow for acceleration or node movement that persist as a layout changes. The result of each 'force' calculation is added to each current node position, and this calculated net force is used to calculate new node positions. If the stopping criterion is met, the algorithm ends, otherwise a new iteration starts.

The resting spring length is the length that all springs in the layout will change to unless acted on by other forces. The resting spring length needs to be calculated at the beginning for each initial layout. The calculation for the resting spring length between nodes is similar for KK and FR; and, they both take the drawing area into account to attempt to ensure the nodes can fit into the scene even if the longest path between nodes in the layout is straight. Because of the similarity between KK and FR, and to attempt to control for any variability that using different resting lengths might have caused, the resting length calculation from KK was used for both algorithms. In this calculation, all link paths are examined to calculate the maximum link path distance (i.e. number of links in the longest path). The resting spring length is calculated by dividing the width of the drawing area (in pixels) by the number of links in the longest path in the layout.

The FR algorithm uses a cooling schedule, but the details are not presented in the source paper. The KK algorithm does not use cooling or friction, but without integer round off to prevent node oscillations, a 'friction' scalar was necessary in my implementation of KK. In Chapter 5, the need for a friction scalar is discussed in detail.

All forces in KK and FR are always scaled by the friction scalar; however, the scalar does not actually change algorithm forces until many cycles past the start. The friction scalar is held at 1.0 (no friction) until after the number of cycles goes beyond a large threshold number of cycles. This threshold is referred to as the friction scalar cycles threshold (fsct), where fsct = 4.5 * $numNodes^2$. As a layout is iteratively adjusted by an algorithm and after fsct cycles have been completed, the friction scalar is multiplied by 99.7%. This compounded scaling/reduction of the friction scalar's magnitude results in a gradual reduction of all algorithm forces over iterations.

4.1.2 Kamada-Kawai Algorithm Specifics

The Kamada-Kawai (KK) algorithm uses three different 'forces' that are applied each time the algorithm is run. The forces affect the movements of the nodes, but only the node that would move the most (the node that is farthest away from being at its local minima) is allowed to move. That node is moved until it has reached its local minimum (i.e. it has reduced its 'energy' down to a negligible amount). All three types of forces on all nodes are calculated each algorithm iteration.

The first type of force calculated in KK is the net spring force. This force is calculated for each node and is the vector sum of all spring forces from all links that are connected to that node. If a node has multiple links because it is directly-linked to multiple adjacent nodes, the spring force calculated is the sum of the spring forces from the links that connect the node to its adjacently neighboring nodes. The second force in KK is for small distance repulsion that acts between any nodes that are very close to one another. The third force is for general node attraction that acts between any and all pairs of nodes. Details of KK implementation are described in the source paper [46].

All of the force calculations result in pixel distance vectors for each node. The vectors (net 'force') are scaled by the friction scalar and then the results are added to each current node position to get new possible pixel locations. At this point no nodes have been moved. It is only the nodes' possible displacements, due to the sum of forces in x and y, that have been calculated. Since KK moves only one node at a time, it may be that no node has been chosen to move yet (e.g. when the algorithm is first started). The node that would move the farthest is chosen because it represents the one that has the highest net force exerted on it. It is the node that is the least at rest. The KK algorithm continues to change the layout one node at a time until the layout reaches the stopping condition at which point the final layout (L′) has been produced.

4.1.3 Fruchterman-Reingold Algorithm Specifics

In the Fruchterman-Reingold (FR) algorithm, two different 'forces' are calculated each time the algorithm is run (each cycle/iteration). As with the KK algorithm, the nodes are moved depending on the net forces experienced by each node. However, in FR, all nodes move each iteration.

The first force calculated in FR is an attraction between linked adjacent nodes. The second force is repulsive and acts between all pairs of nodes. The friction scalar is applied to all forces, and then the displacement due to this force is applied to the node positions. All forces on all nodes are calculated each algorithm iteration. Details of FR implementation are described in the source paper [36].

4.2 Calculating Statistics

A collection of layouts is referred to as a set. For example, the set of initial layouts of size 15 is denoted as $S(L_{15})$. In the experiment, to control for initial conditions, the same initial layout set (S(L)) is used for both algorithms, resulting in two different final layout sets ($S(L'^{KK})$ and $S(L'^{FR})$) for each graph size (15 or 30). Thereby, this experiment produces four sets of final layouts: $S(L'^{KK}_{15})$, $S(L'^{FR}_{15})$, $S(L'^{KK}_{30})$, and $S(L'^{FR}_{30})$. Application of graph layout algorithms (KK and FR) to many initial layouts (S(L)) represents many parallel experimental trials. The resulting pairs of results can be statistically compared using 2-tailed paired t-tests (discussed below).

The layouts for a particular graph size (15 or 30 nodes) are scaled such that their average link lengths for all layouts of that graph size are the same. This adjustment causes $S(L'^{KK}_{15})$ and $S(L'^{FR}_{15})$ to have the same average of average link lengths as one another (as will $S(L'^{KK}_{30})$ and $S(L'^{FR}_{30})$). Scaling graph layout sizes based on average link lengths normalizes the size of the layouts for each set of trial runs. Normalization is accomplished by calculating the average of the average link lengths for all KK and FR trial runs for a given graph size. The averages are compared and all KK average link lengths, average node displacements, and average activity measures are scaled (by the same amount) until the average of the average link lengths are the same for KK and FR.

After the graph layout sizes have been normalized for all pairs of KK and FR layouts (at both graph sizes separately), the remaining measurements (described in Chapter 3) and statistics (described in 4.2.1 to 4.2.8) can be calculated. Average and standard deviation are calculated from the measurements, and the results produce a statistical characterization of an algorithm's behaviour under the conditions that were used. The statistics calculations are performed on the 15 node trial runs ($S(L'^{KK}_{15})$ and $S(L'^{FR}_{15})$) separately from the 30 node trial runs ($S(L'^{KK}_{30})$ and $S(L'^{FR}_{30})$).

To compare the experimental results, 2-tailed paired t-tests are performed to assay if there is a statistically significant difference between the pairs of sets ($S(L'^{KK}_{15})$ versus $S(L'^{FR}_{15})$, and $S(L'^{KK}_{30})$ versus $S(L'^{FR}_{30})$). To test if there is a difference between the averages from the two sets of samples, it does not matter which sets of values are bigger or smaller - it only matters that there is a difference that is statistically significant. Rejecting the null hypothesis in a 2-tailed t-test means there is a significant difference between the averages, where the average of one set is above or below the other's set. This is why the test is called a 2-tailed t-test (2 alpha regions: one positive and one negative).

A paired t-test uses pairs of results and it has more statistical power since the effects of any potential confounding factors (e.g. noise in the data) are reduced. Paired t-tests can be used when the differences between the pairs have an approximate normal distribution; so, histograms are created for each set of difference measurements at each graph size, and the difference values are observed to ensure they are approximately normally distributed. This verification of normal distribution (approximately bell shaped) was accomplished by visually comparing histograms from the data to the corresponding normal probability curve. An example comparison is shown in Figure 4.2.1.

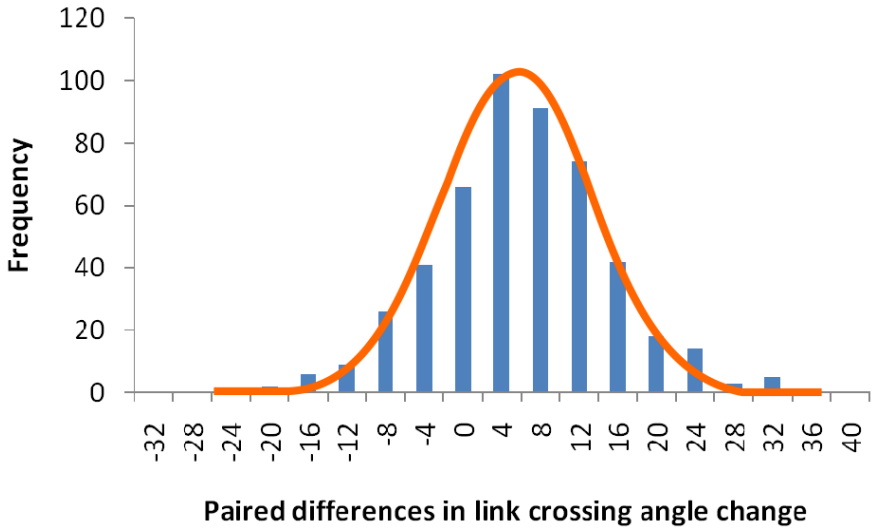


Figure 4.2.1: Histogram of difference in link crossing angle change (KK-FR) data pairs for graph size 30 showing approximate normal distribution (shown by the orange curve). The shape of this histogram is a representative example of the distributions of each data set for the t-tests performed.

A common tactic used in various research fields (including graph drawing) is to use many trials to increase the statistical significance of the results [44][12][88][51][5][77], and this can be accomplished by increasing the "degrees of freedom" (df). The df for a paired t-test is N-1, where N is the number of pairs (i.e. the number of trial runs performed). Since, 502 runs (N=502) are used in the experiment, the df is at least 500, so the statistical power of the t-tests is very high (501 > 500). Any df above 500 is taken as a df of infinity, and the critical t-value is as low as possible for a given p-value.

The critical t-value is a threshold value that the t-value (resulting from a t-test) is compared against. In terms of statistical significance, a larger critical t-value represents a high significance level used in the t- test. A statistically significant difference is found between paired data sets if the absolute value of the t-value is greater than the critical t-value. This result would mean the null hypothesis can be rejected and the difference between the two sample sets (e.g. for a

particular measurement of KK and FR layouts at a particular graph size) are statistically significant.

The critical t- value is obtained from a statistics table of critical t-values based on the values of the p-value and df. I am using a p-value of p=0.05, which means that, if the absolute value of the t-value calculated is greater than the absolute value of the critical t-value, there is less than a 5% probability that the results will have happened by chance. The 2-tailed critical t-value for p=0.05 and df>=500 is 1.960.

4.2.1 Average and Standard Deviation for Time to Completion

The average and standard deviation are calculated for the time an algorithm takes to complete each trial run.

4.2.2 Average and Standard Deviation for Number of Algorithm Cycles Per Second

The average and standard deviation are calculated for number of algorithm cycles per trial. The number of cycles per second is calculated by dividing the number of cycles to completion by the time to completion. This calculation is done for each trial, and from these values, the average and standard deviation are calculated for the number of cycles per second.

4.2.3 Average and Standard Deviation for Average Node Displacement

The average node displacement is calculated for each trial run. The displacement is the net change in a node's position (measured in pixels) from initial to final layout. The average node displacement for a layout is a sum of the absolute values of each node's displacement divided by the number of nodes. From the average node displacement values from all trial runs, the average and standard deviation for the average node displacement are calculated.

4.2.4 Average and Standard Deviation for Average Node Activity

The average node activity is calculated for each trial run. The node activity is a node's total travel distance from initial layout through all layout steps to the final layout. This measurement does not consider direction (unlike displacement). From the average node activity values from all trial runs, the average and standard deviation for the average node activity are calculated.

4.2.5 Average and Standard Deviation for Average Delta Directly-Linked-Orthogonal Ordering (DDLOO)

The Delta Directly-Linked Orthogonal Ordering (DDLOO) is calculated between initial and final layouts. The average DDLOO for a layout is calculated by dividing DDLOO by the number of nodes in the layout. The average of the average DDLOO is the sum of the average DDLOO values for all trials divided by the number of trials.

4.2.6 Average and Standard Deviation for Average Link length

The average link length is calculated for each trial run. From these values, the average and standard deviation of average link lengths is calculated for all trial runs.

4.2.7 Average and Standard Deviation for Number of Link Crossings

The average and standard deviation are calculated for the number of link crossings in the initial layouts and in the final layouts.

4.2.8 Average and Standard Deviation for Average Link Crossing Angle

The average link crossing angles are calculated in each initial and final layout. The average and standard deviation are calculated for the averages in the initial layouts and in the final layouts.

5. Results and Discussion

5.1 Results and Discussion of the Experiment

The measurements described in Chapter 3.2 were used to compare two well-known Force-Directed layout algorithms (Kamada & Kawai (KK), and Fruchterman & Reingold (FR)). Averages and standard deviations on these measurements (described in section 4.2) were computed and used to characterize and find behavioural differences between the algorithms as a basic proof of concept for the usefulness of the measurements. The mental map is not measured or tested in this study; however, the methodology described in this thesis is concerned with using measurements and calculating statistics to characterize layout adjustment algorithms in terms of how well the algorithms may be able to maintain a user's mental map. The results of the statistics calculations from the experiment for treatments KK and FR on graph sizes 15 or 30 are presented in this chapter; and, Figures 5.1.8.1 to 5.1.8.8 correspond to the measurements described in Chapter 3.

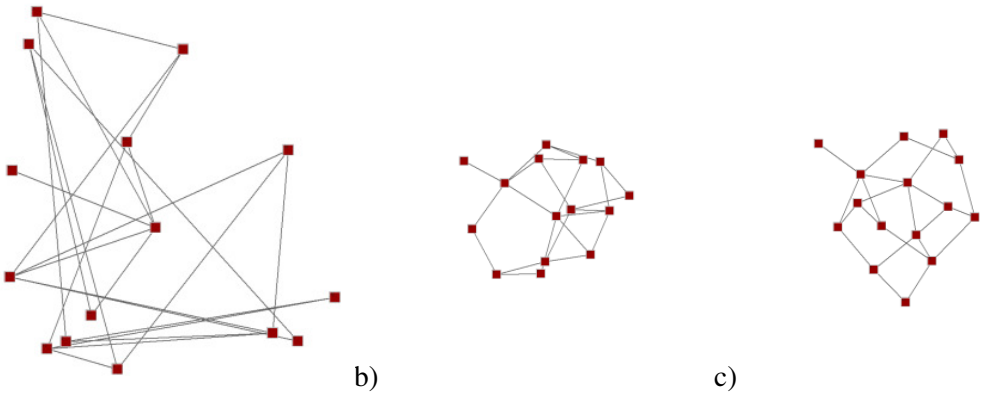
5.1.1 Two Similar and Well-Known FD Layout Algorithms Were Used for a Basic Proof of Concept

Two similar algorithms were used to validate the usefulness of the measurements presented in this thesis instead of using several vastly different algorithms because I believed that working with similar algorithms would lead to more relevant and interesting results (while still tightly controllable). Furthermore, for the purposes of my study, two similar algorithms that are well established in the research community [67] are sufficient. The Kamada-Kawai (KK) [46] and Fruchterman-Reingold (FR) [36] algorithms were chosen for this study because both have a number of important similarities in terms of functionality and requirements which allow the experiment to be tightly controlled. Neither KK nor FR consider link crossings, neither considers planarity [16], both use undirected general graphs (links do not have directionality, and there is no inherent structure) [44], both use straight (non-curved) links, both can use the same range of graph sizes (number of nodes and links), the input graphs for both can be random (unlike planar

graph drawing procedures [22]), linkage densities (number of links per node in a graph) can be in the same ranges [46][36], and neither algorithm explicitly prevents node occlusions or avoids link crossings [36][85]. The algorithms are FD algorithms derived from Eades' Spring Embedder [46][36], and both use pseudo-forces based on node positions to iteratively adjust a layout by moving nodes. Both algorithms produce a final layout by using 'energy' reduction/minimization (optimization) to find a stable solution (i.e. a final layout) [43][85]. These factors make KK and FR good choices for doing a comparison using the measurements and statistics presented in this thesis, as a basic proof of concept.

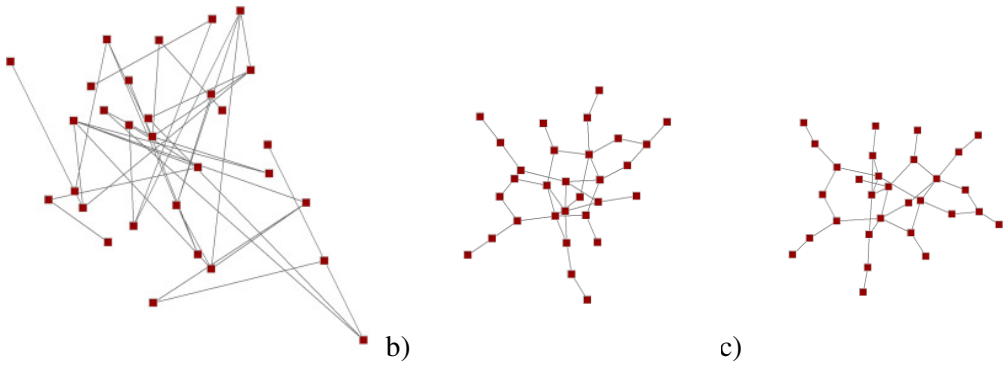
It is important when comparing algorithm behaviour that both algorithms are capable of taking the same type of initial graph layout as input [7] because, while initial node positions do not have a substantial affect on the resulting drawings, the initial node positions will affect the number of iterations required to reach a final layout [33]. Using random initial layouts is appropriate for this study since it has been suggested that the use of random initial layouts is important to ensure algorithm performance is evaluated from a practical point of view [21]. Note that the use of random initial layouts as a starting point in the experimentation does not mean the initial layouts necessarily have anything of practical worth that a user would want to build a mental map of; but, using a large number of random initial layouts is an effective method for statistically evaluating if an algorithm tends to improve a layout with regards to specific measurements (and the related mental map preservation principles). Random layouts have been suggested as representative of structural characteristics that arise in many practical applications [21]. Most FD algorithms start with random initial layouts [37][56][67], and random layouts are commonly used to compare the performance of layout algorithms [21][7][12][24].

Each trial started with the generation of an initial random graph layout (L), and subsequently the graph layout algorithm in question was applied to produce a final layout (L′). In terms of the nomenclature used in this thesis, if KK was used on a layout of size 15, it would result in a final layout referred to as L'^{KK}_{15}. The same starting data was used for both KK and FR, so one layout (L) would be used to produce a pair of resulting final layouts (L'^{KK} and L'^{FR}). Figures 5.1.1.1 and 5.1.1.2 are screenshots showing typical initial and final layouts at graph sizes 15 and 30 respectively.



a) b) c)

Figure 5.1.1.1: Screenshots of initial and final layouts for a size 15 graph. Figure (a) shows the random initial layout with a 1.53 link density. Figures (b) and (c) are the final layouts that resulted from KK and FR treatments respectively. The KK algorithm took 1066 cycles and FR took 169 cycles.



a) b) c)

Figure 5.1.1.2: Screenshots of initial and final layouts for a size 30 graph. Figure (a) shows the random initial layout with a 1.2 link density. Figures (b) and (c) are the final layouts that resulted from KK and FR treatments respectively. The KK algorithm took 4254 cycles and FR required 448 cycles.

5.1.2 Precision

Both algorithm implementations use floating point precision. Precision is particularly important for node positions to avoid integer arithmetic issues. These problems may include round-off errors (a behaviour artifact) that might cause nodes to block-up [36], or migrate towards the top-left [44], or potentially cause unknown affects [33] because of value truncations. Precision problems associated with integer round-off may have been found in KK [46], FR [36], GraphEd [44][7], and other research that uses integer arithmetic (e.g. [33]). With even a basic modern computer (as compared to those from 20 years ago), a programmer has the luxury of using floating point precision without sacrificing algorithm speed. In newer computers, the use of integer arithmetic to speed up computations is not an important benefit as long as the number of nodes is kept relatively low (e.g. 30 nodes). Floating point precision allows my implementation to avoid the use of work-around heuristics like having to turn off repulsive forces every 5 cycles to avoid the blocking affect [36].

5.1.3 Graph Sizes

Graph sizes 15 and 30 were used in this study for four key reasons. Approximately this number of nodes were used in important figures in both [46] (Figure 2) and [36] (Figures 8 and 9) to demonstrate the algorithms' specific behaviours. Both 15 and 30 node graph sizes are considered small by today's standards, and they are both reasonable sizes for applying Force-Directed algorithms [33][48][67][17]. Keeping the graph sizes small was desirable because it is well known that user comprehension, and therefore preservation of the mental map, in graph structures is easiest if the graph size is small enough to work with [43][9]. Both graph sizes are similar to graph sizes used in graph perception research that tries to avoid cognitive burden, information overload, or creating graph drawings that are visually incomprehensible (i.e. fewer than 30 nodes) [71][87][32][68].

5.1.4 Common Initial Layout Attributes for KK and FR

KK and FR algorithms both operate on layouts that have common requirements. Layout attributes that are needed for both algorithms include straight line links, undirected links, connected graphs (all nodes are at least indirectly linked to one another), random node placement, nodes are treated as points with no size, random linkages, and the link density is within a certain range [46][36][33][7][24][57][27]. A square graph drawing area (e.g. 1000x1000 pixels) was used because it is a criterion of KK [46]. The number of nodes per area used in this study is similar to [87]. The link density range (1.2 to 1.6 links per node) was used because this falls in the middle of what is considered to be the normal range [67]. Also, this range is consistent with the approximate range of link densities shown in both KK and FR source papers [46][36].

5.1.5 The Stopping Condition

Each algorithm was allowed to continue running each trial until a specific stopping criterion was reached. Typical completion criteria include number of cycles or the reduction of energy, but it is not easy to determine stopping conditions from the literature for either algorithm [36]. For this research, it was important to choose a stopping condition that treated the algorithms equally fairly. It was important to ensure that both algorithms were each given a sufficient number of cycles to arrive at layout solutions, but it was also important to prevent them from running indefinitely if they were essentially already at a solution but still producing small node movements per cycle (i.e. insignificant oscillations that did not lead to any significant layout change). I decided to use an energy threshold because the number of cycles sometimes varied considerably from run to run. If I used the number of cycles as a threshold for the stopping condition, this stopping condition would not necessarily be related to the algorithm finding a solution (a final layout). A stopping condition based on a threshold number of cycles has obvious problems if the algorithm does not have enough cycles for convergence on a solution, or if the algorithm wastes cycles (e.g. oscillating without any significant layout change) until the threshold is hit [33]. Furthermore, because the algorithms (KK and FR) are fundamentally related to a notion that the ‘energy’ in a system (i.e. the layout’s forces which is directly related to node

movement) is being reduced as the layout iteratively becomes the final layout, it makes sense that the stopping condition would be directly related to the layout's propensity for node movement.

In this experiment, an algorithm is allowed to continue adjusting a layout until there is a negligible amount of change in node movements in a cycle. In my research, this negligible change was defined as less than 2 pixels of total node movement in a layout per cycle (i.e. sum of net movements from all nodes in the graph in one algorithm iteration). By trial and error, this stopping value threshold produced results for both algorithms that did not have considerable residual energy causing nodes to bump around incessantly near the end of each run (wasted algorithm cycles). Conversely, if the threshold value was large (e.g. 5 or 10 pixels), a layout would not be able to unravel itself because it would often get stopped when migrating nodes were trying to move through tight gaps formed by other nodes. In the 1000x1000 pixel layout area used, less than 2 pixels of total movement, by all nodes, represents a very small amount (merely 0.2% in one dimension). Stopping a layout algorithm when it produces less than this amount of node movement is analogous to the algorithm finding a layout solution in which the energy is significantly minimized. Therefore, a node movement based ('energy' based) stopping condition was used because, of the two possible types of stopping conditions (threshold number of cycles or node movement based), using a node movement threshold was considered the most fair to both algorithms.

Note that the stopping condition's energy metaphor in this experiment is related to kinetic energy - not potential energy. For instance, a layout may find a solution in which all nodes are at their local minima (i.e. they are not moving), but a lack of node movement does not mean the total energy in the system is optimally minimized. It simply means the kinetic energy in the system is at a minimum. The FD graph drawing literature typically refers to energy in this manner [64].

5.1.6 Controlling the Experiment

In each trial run, the number of nodes and graph connectivity (links between nodes) was controlled by holding them constant because altering the underlying graph data (i.e. changing the number of nodes or adding/subtracting links) may affect the mental map [27][60][32]. The Force-Directed layout algorithms that were chosen for comparison by the measurements/statistics presented in this study change node positions after the information space has already been

initially populated (i.e. no nodes or links should be added or subtracted during each experimental run). Maintaining the number of nodes and the specific links was important to control for layout changes that are not related directly to typical layout adjustment algorithm behaviours.

During implementation and testing of KK and FR, the layouts were drawn as they changed (at each cycle from L to L′), but to attempt to remove any hardware-related time delays that might have been caused by drawing to the screen, the entire experiment was done without drawing any layouts. Also, to ensure that the statistical calculations did not affect the experiment, the statistics were only done once the code was no longer timing each trial run.

The geometric layout sizes for all trial runs was controlled in the experiment so that the distance related measurements and statistics (i.e. Figures 5.1.8.3, 5.1.8.4, and 5.1.8.6) would not be biased due to the variation between the way forces are treated in KK and FR. To control for geometric layout size in final layouts ($S(L'^{KK}_{15})$ versus $S(L'^{FR}_{15})$, or $S(L'^{KK}_{30})$ versus $S(L'^{FR}_{30})$), a scalar was calculated at runtime for each final layout and used to resize each L'^{KK} such that $S(L'^{KK})$ had the same average link length as $S(L'^{FR})$ for each graph size tested. Controlling for the layout size factor by normalizing the link lengths allows these statistics to be compared between KK and FR results without bias due to layout size. If the layout sizes were not normalized, the measurements and statistics that depend on node positions (average of the average displacement per node, and average of the average activity per node) would be biased by the fact that the layouts were not the same size. The scalar value for graphs of size 15 was 0.9477 (95%), and for size 30, the scalar was 0.9998 (essentially 100%). Normalizing for graph size in this way is the same approach as used by Frick [33].

5.1.7 The Friction Scalar

In addition to the need to control the experiment in several ways to make comparisons between KK and FR valid, it was also necessary to add a 'friction' scalar. To ensure small node movements would not keep the algorithms oscillating indefinitely, after many cycles (many algorithm iterations) a gradually changing friction scalar was implemented. This approach is similar to the cooling schedule idea from [36]. The friction scalar was applied to all forces equally in a particular algorithm to gradually reduce the magnitudes. Note that this 'friction' is

not a true force, but it is an analogous counteracting 'force' which reduces other forces' magnitudes. Reducing the effect of other forces has the direct effect of reducing node movements since forces in KK and FR directly relate to node movement.

To give an algorithm a long time to work before interfering (i.e. applying the scalar), the friction was only applied after a certain number of cycles relative to the size of the graph. After many test runs, it was concluded that both algorithms had more than enough cycles to reach a final layout after friction scalar cycle threshold (fsct) cycles, where fsct = 4.5 * $numNodes^2$. After this threshold, the friction scalar was applied to prevent layouts from oscillating (without significant change to the layout). It was necessary to make the threshold number of cycles exponentially relative to the size of the graph because larger graphs take significantly longer to complete.

The friction scalar was initially set to 1 (i.e. 100%). For each cycle past the threshold, the scalar was multiplied by 99.7%. The compounded scaling by 99.7% slowly reduced the friction scalar's value which gradually lowered all the forces (used in both the KK and FR algorithms). This amount of reduction per cycle (0.3%) was found to be reasonable because it slowly reduced the oscillations in a non-abrupt manner. After the algorithm had many cycles to act, uninhibited by friction, the gradually reduced force magnitudes produced progressively smaller node movements (especially oscillations in KK). Thereby, the slow reduction of forces preserved node movements for many cycles past the threshold (fsct), and eventually the nodes smoothly found positions of local minima.

Node movement eventually stopped because the slow stepwise increase in friction caused the forces to be reduced to a point where they could no longer move the nodes. This friction scalar was absolutely necessary for KK to be able to stop using the stopping condition for graph size 30. Without this friction scalar, KK would often oscillate seemingly indefinitely. This problem was probably exacerbated because the algorithm only allows one node to move per algorithm cycle. These oscillations ruined KK's chance to find a final layout even though the nodes had been oscillating by only a few pixels around the same positions for tens of thousands of cycles. A small amount of node movement (>2 pixels of movement in a layout) was enough to prevent the layout from reaching the stopping condition. In some cases, KK was able to find a layout solution before the friction would be engaged, but in many cases, friction was required to allow KK to reach a layout solution just past the fsct number of cycles. Without the small amount of

friction applied, the experiment would have required a fixed number of maximum cycles, and KK would have been severely outperformed by FR.

Figures 5.1.1.1 and 5.1.1.2 demonstrate the necessity for friction. For example, in Figure 5.1.1.2, KK required 4254 cycles to complete; however it was observed that the layout was essentially complete at 1700 cycles. Node movements after this point were small oscillations that did not result in any further significant changes to the layout. During testing, this oscillation phenomenon prevented KK from stopping in many layouts (even after tens of thousands of cycles). Friction was required to calm the node oscillations to allow the algorithm to stop. However, if friction was used too early, the algorithm may not have had a chance to run to completion. Because of the complexity of algorithm behaviour, if a layout does not seem inclined to stop (e.g. KK), it is difficult to establish exactly at what point friction can or should be applied. Therefore, to allow an algorithm to have a chance to stop at a layout solution, a reasonable heuristic was to allow KK to run for at least twice the number of cycles it consistently seemed to need to become essentially done (i.e. $fsct_{30} = 4.5 * 30^2 = 4050 > 2 * 1700$). Note that this rationale is difficult to prove, but friction was necessary to get results for KK, measure the other algorithm behaviours, and to conduct the experiment.

5.1.8 Results

The results of the experiment are shown in Figures 5.1.8.1 to 5.1.8.8. The figures below show averages for KK and FR algorithms applied to graph sizes with 15 and 30 nodes. The error bars show standard deviations.

Figure 5.1.8.1 clearly shows that Fruchterman and Reingold's algorithm significantly outperforms Kamada-Kawai's algorithm in terms of speed, and Figure 5.1.8.2a shows that FR requires fewer cycles to complete. Figure 5.1.8.2b shows that FR, when unencumbered by drawing to the screen, has the capacity to produce a higher frame rate than KK at both graph sizes (18290 > 2577 and 5602 > 668 cycles per second). Even if encumbered by the lag time involved with drawing a layout to the screen, FR stands to produce 7.1 times more frames per second at size 15, and 8.4 times more frames per second at size 30. From another perspective, dividing the average time to completion by the average number of algorithm cycles shows that FR is 6.6 times faster per cycle for layout size 15, and 8.4 times faster per cycle than KK at layout size 30.

In terms of mental map preservation, minimizing the time to completion is advantageous; however, the measurement is dependent on the computer on which the experiment is run. The hardware dependence of the time measurement makes layout adjustment speed comparisons unreasonable unless the experiments are done on the same computer setup [33].

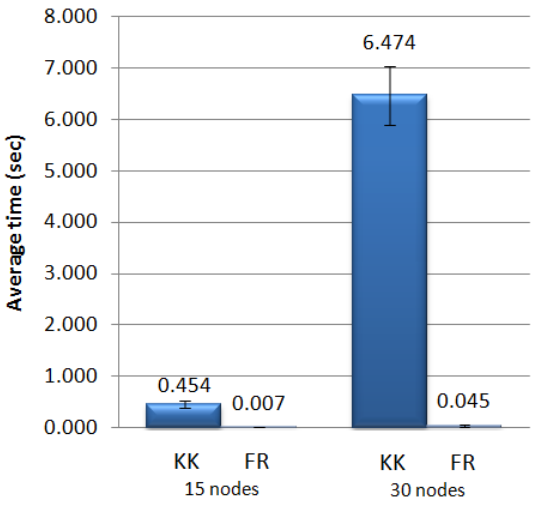
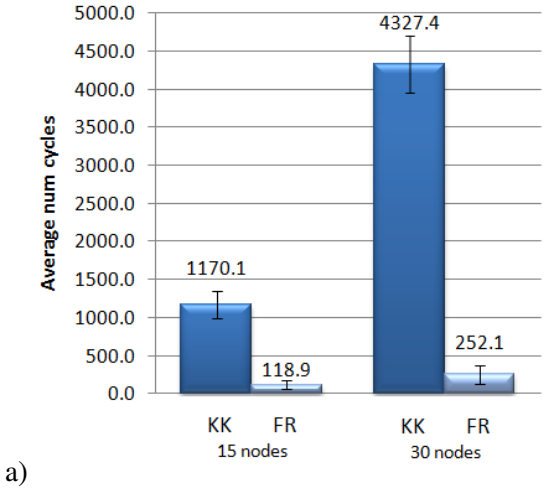
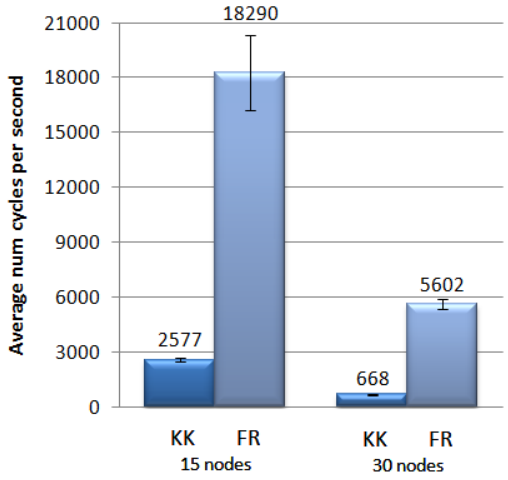


Figure 5.1.8.1: Average time (sec) per trial. In terms of the mental map preservation principles, lower times are preferable. The FR algorithm drastically outperforms KK in terms of speed for both size 15 and 30. The standard deviations for $S(L'^{KK}_{15})$, $S(L'^{FR}_{15})$, $S(L'^{KK}_{30})$, and $S(L'^{FR}_{30})$ are 0.072, 0.003, 0.570, and 0.020 respectively. The t-values for sizes 15 and 30 are -229.45 and -199.06 respectively.



a)



b)

Figure 5.1.8.2: Average number of cycles per trial and average number of cycles per second per trial. Figure (a) shows the FR algorithm uses substantially fewer cycles to run than KK. Note that the KK algorithm moves only one node per algorithm cycle, whereas FR moves all nodes each cycle. Figure (b) shows that FR produces more cycles per second and therefore more layout adjustment steps per second. For (b), in terms of mental map preservation, higher cycles per second values are preferable. For number of cycles, the standard deviations for S(L'^{KK}_{15}), S(L'^{FR}_{15}), S(L'^{KK}_{30}), and S(L'^{FR}_{30}) are 174.2, 51.8, 379.6, and 122.3 respectively. The t-values for sizes 15 and 30 are -220.09 and -190.26 for graph sizes 15 and 30 respectively. For cycles per second, the standard deviations are 116, 2040, 12, and 287; and, the t-values are 254.83 and 571.65.

Not only is the number of cycles useful for comparing an algorithm's cycles per second, but the measurement is also useful for examining how the friction scalar played a role in the experiment. Figure 5.1.8.2a shows that at graph size 15, the average number of cycles required for KK was 1170 with the standard deviation reaching down to about 1000 cycles ($fsct_{15}$ is 1013). At graph size 30, $fsct_{30}$ is 4050 and the average number of cycles required for KK was 4327 with the standard deviation extending down to just under 4000 cycles. These results lend credibility to the necessity of a friction scalar in aiding the KK algorithm. Notably, the friction scalar was seldom needed for FR because this algorithm completed well before the threshold (e.g. $118 < 1013$ and $252 < 4050$). Because the threshold number of cycles and friction scalar were necessary for KK but not for FR, this implementation feature is a significant problem when considering time per trial run and number of cycles per assay.

In terms of node displacement (Figure 5.1.8.3), the averages at size 30 are similar to the point that the 2-tailed paired t-test cannot conclude they are significantly different. In the same figure, we see that at size 15, the displacement values indicate that KK outperforms FR, and the activity values (Figure 5.1.8.4) also support the idea that KK is more efficient at adjusting graphs of size 15. However, although FR has the same displacement as KK at size 30, FR is considerably more efficient at moving nodes (i.e. less activity) to their final resting places at this size and will produce more algorithm cycles per second, so the same displacement would appear smoother over more frames per second with FR.

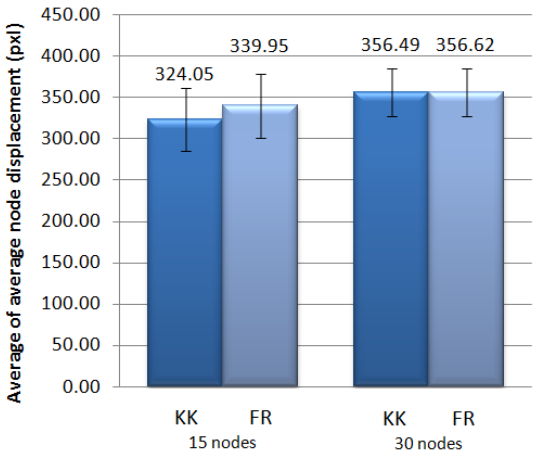


Figure 5.1.8.3: Average of average node displacement (pixels) per trial. In terms of the mental map preservation principles, lower displacement values are preferable. The standard deviations for S(L'^{KK}_{15}), S(L'^{FR}_{15}), S(L'^{KK}_{30}), and S(L'^{FR}_{30}) are 37.99, 38.71, 29.36, and 28.82 respectively. The t-values calculated for this measurement are 109.67 and 65.58 for graph sizes 15 and 30 respectively.

The DDLOO statistics (Figure 5.1.8.5) show that although there is considerable overlap in values, FR generally produces less shape change than KK at both graph sizes which means that FR may be slightly better at preserving a user's mental map. When choosing between KK and FR, an algorithm designer may not want to use this statistic as a strong deciding factor. However, examining the combination of results may be more helpful. For instance, it is interesting that KK and FR result in a similar amount of displacement at both graph sizes (Figure 5.1.8.3), but FR still manages to more quickly and in substantially fewer cycles (i.e. more efficiently) (Figures 5.1.8.1 and 5.1.8.2a) arrive at a layout solution while producing less shape change (Figure 5.1.8.5). In terms of mental map preservation, it is also interesting to note that FR has a higher number of cycles per second (Figure 5.1.8.2b), and therefore the displacement, activity, and local shape change is not only being produced in less time for FR, but also with potentially a higher frame rate (i.e. algorithm iterations per second), which would display the layout changes more smoothly over time. This higher frame rate animation with the FR algorithm would allow a user to view a smoother transition from L to L′ and this may help them preserve their mental map [34]. In terms of the mental map, animation smoothness, and the drive to minimize layout changes between

algorithm iterations, the comparison of FR to KK algorithm cycles per second shows that FR produces 6.6 times more frames per second at size 15 and 8.4 times more frames per second at graph size 30.

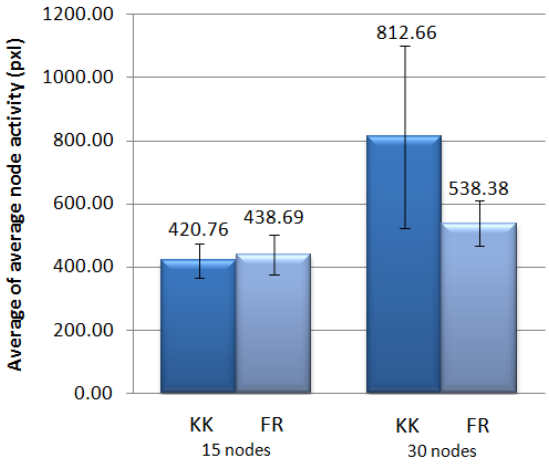


Figure 5.1.8.4: Average of average activity per node (pixels) per trial. In terms of the mental map preservation principles, lower activity values are preferable. The standard deviations for $S(L'^{KK}_{15})$, $S(L'^{FR}_{15})$, $S(L'^{KK}_{30})$, and $S(L'^{FR}_{30})$ are 54.17, 62.92, 290.10, and 71.59 respectively. The t-values calculated for this measurement are 68.10 and 35.26 for graph sizes 15 and 30 respectively.

In terms of link lengths, my results (Figure 5.1.8.6) agree that both KK and FR produce similar link length distributions. This result was found by Himsolt [44], and Klemetti says that spring algorithms usually result in similar link lengths [48]. The fact that the normalization steps (at size 15 and 30) used scalars that were close to 1 (95% and 100%) shows that even if the results were not normalized for layout size, the link lengths were still very similar between KK and FR results.

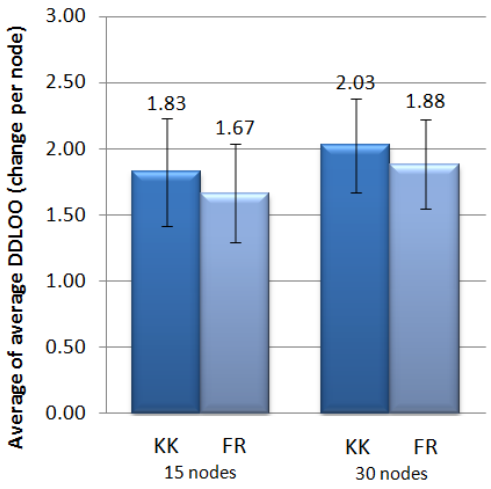


Figure 5.1.8.5: Average of average Delta Directly-Linked Orthogonal Ordering (average DDLOO) per trial. In terms of the mental map preservation principles, less change per node is preferable. The DDLOO is a measure of graph layout shape change and is explained in Chapter 3 (see Section 3.2.5 and Figure 3.2.6.1). The average DDLOO is simply the DDLOO value for a layout divided by the number of nodes. The average of average DDLOO is the average DDLOO over all trial runs in the experiment. The FR algorithm produces less graph layout shape change for both graph sizes. The standard deviations for $S(L'^{KK}_{15})$, $S(L'^{FR}_{15})$, $S(L'^{KK}_{30})$, and $S(L'^{FR}_{30})$ are 0.41, 0.37, 0.36, and 0.34 respectively. The t-values calculated for this measurement are -9.08 and -13.03 for graph sizes 15 and 30 respectively.

Because it was necessary to control for the geometric size of the layouts, the link length statistics cannot give useful information to help a designer decide what algorithm is best in terms of minimizing link lengths. Also, since the standard deviations for the link lengths are essentially the same for size 15 and 30 for both KK and FR, choosing between the algorithms based on link length variation is not possible.

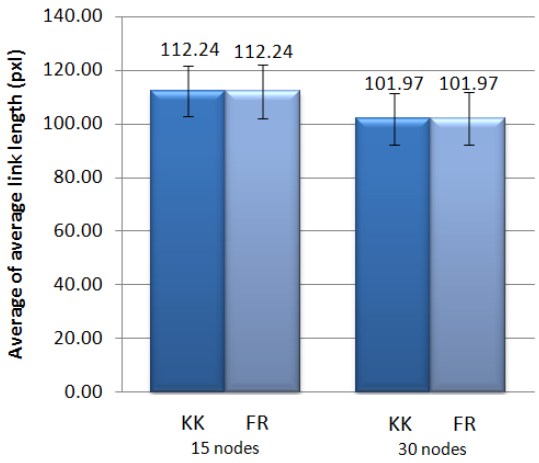
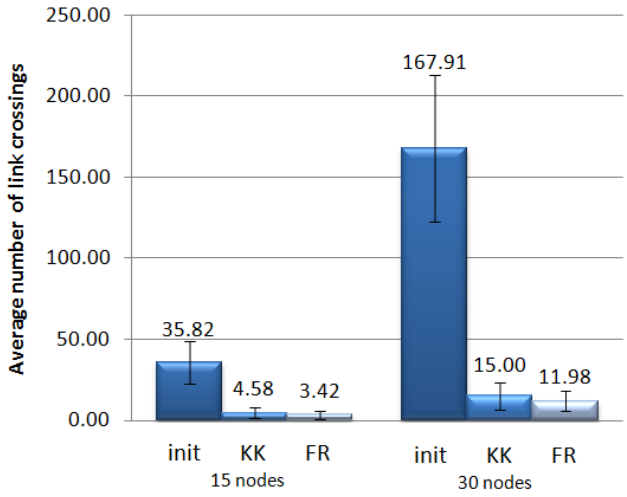


Figure 5.1.8.6: Average of average link length (pixels) per trial. In terms of the mental map preservation principles, smaller link length and standard deviation values are preferable. The standard deviations for S(L'^{KK}_{15}), S(L'^{FR}_{15}), S(L'^{KK}_{30}), and S(L'^{FR}_{30}) are 9.38, 10.01, 9.55, and 9.83 respectively. The t-values calculated for this measurement are -0.026 and -0.030 for graph sizes 15 and 30 respectively.

Figure 5.1.8.7 shows that random layouts used for initial layouts resulted in many link crossings. As expected, the number of link crossings is substantially higher when the number of nodes is doubled. Both KK and FR were able to considerably reduce the number of link crossings for both graph sizes tested, however, paired t-tests show that FR significantly outperforms KK at both graph sizes. Results suggest that FD is generally better than KK at improving graph layout and reducing visual clutter with respect to minimizing the number of link crossings.

a)



b)

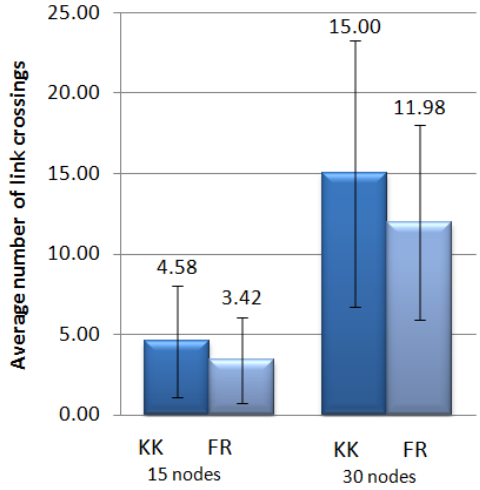


Figure 5.1.8.7: Average number of link crossings per trial. Figure (a) shows average number of link crossings for initial layouts, final KK layouts, and final FR layouts. Figure (b) is an expanded view of average number of link crossings for final KK and FR layouts. In terms of the mental map preservation principles, number of link crossings closer to 0 is preferable. The standard deviations for $S(L_{15})$, $S(L'^{KK}_{15})$, $S(L'^{FR}_{15})$, $S(L_{30})$, $S(L'^{KK}_{30})$, and $S(L'^{FR}_{30})$ are 13.18, 3.48, 2.66, 45.05, 8.31, 6.04 respectively. The t-values calculated for this measurement are -19.78 and -14.28 for graph sizes 15 and 30 respectively.

At graph size 15, both algorithms were occasionally able to find layout solutions with no link crossings. However, at graph size 30, neither KK nor FR produced layouts with no link crossings. The fact that no final layouts at size 30 resulted in zero link crossings may be an inherent issue with the construction of the initial layouts.

Figure 5.1.8.8 shows that both sets of initial layouts for both graph size 15 and 30 produce link crossing angles that are far from the optimal 90 degrees. As with number of link crossings, although KK and FR are not designed to target small link crossing angles, they both do a respectable job of increasing the average link crossing angle. The KK and FR results for this statistic show considerable overlap, but FR generally results in slightly higher link crossing angles.

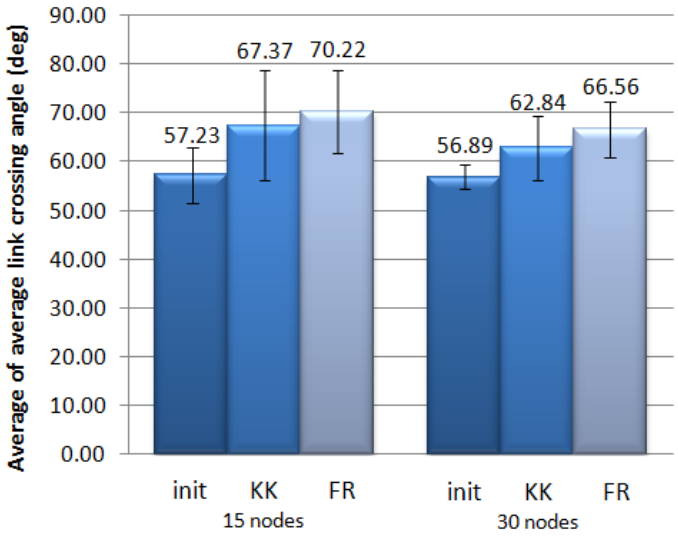


Figure 5.1.8.8: Average of average link crossing angle (degrees) for initial layouts, final KK layouts, and final FR layouts per trial. In terms of the mental map preservation principles, link crossing angles closer to 90 degrees are preferable. The standard deviations for $S(L_{15})$, $S(L'^{KK}_{15})$, $S(L'^{FR}_{15})$, $S(L_{30})$, $S(L'^{KK}_{30})$, and $S(L'^{FR}_{30})$ are 5.70, 11.32, 8.49, 2.55, 6.55, 5.80 respectively. The t-values calculated for this measurement are 8.24 and 15.53 for graph sizes 15 and 30 respectively.

Note that average number of link crossings and average link crossing angles could have been measured as differences between L and L′. However, these measurements were not treated as differences because, in each case, clear optimal values are available for comparison. For average number of link crossings, the optimal value is 0, and for average link crossing angle, the optimal value is 90 degrees.

To check if the differences in averages were statistically significant, 2-tailed paired t-tests were used. All averages were found to be significantly different with two exceptions. The first exception, which was not expected, was the node displacement averages for graph size 30 (see Figure 5.1.8.3). The second exception was the average of average link length statistic (see Figure 5.1.8.6), which was expected to be the same between KK and FR, because the average link lengths were normalized for each set of layouts at each graph size.

The FR algorithm outperforms KK in many ways for graph layouts tested at 15 and 30 nodes. In terms of speed and number of cycles for algorithm completion, FR is much faster and more efficient. In terms of cycles per second, FR produces a substantially higher rate. Both algorithms show about the same amount of displacement from initial to final layouts, but at 30 nodes, FR arrives at a solution with much less activity. In terms of layout shape and orthogonal ordering, FR produces less change. In terms of number of link crossings and link crossing angles, FR is better at reducing visual clutter. With the exception of average of average displacement for graph size 30, 2-tailed paired t-tests confirm significant differences between the statistics. Many trial runs were used to ensure the statistics are sound [51][5][44]. For all statistics, results were found to be significant at a p-value of less than 0.001. These statistical results show there is less than a 0.1% probability that the results are due to chance. Therefore, in terms of the mental map related statistics presented in this research, it can be said that FR is probably better at preserving a user's mental map at both graph sizes. Therefore, if a graph system or layout adjustment algorithm designer is trying to decide between Kamada-Kawai and Fruchterman-Reingold for use on graph sizes around 15 or 30 nodes, choosing FR is a safer choice in terms of preserving their user's mental map.

5.1.9 Comparing These Results to Other Research

The work reported in this study is comparable to [44] and [33]. In comparing run times to [44], my results agree that KK is rather slow. There is a large speed difference between KK and FR; whereas, Himsolt found that both algorithms were slow (by 1995 standards). In my implementation, FR is 65 times faster than KK at size 15 and 144 times faster at size 30. The fact that FR is from one to two orders of magnitude faster than KK suggests that the two algorithms are not in the same speed categories. My time results were more similar to those presented by Frick (1994) in which FR was found to be faster, and KK was considerably slower when used with larger graphs.

Brandenburg used an implementation of KK that was tuned to go faster as compared to Frick or Himsolt's work [7]. The exact way the algorithm was sped up is not known or described, so comparing results is, unfortunately, not possible or reasonable.

5.1.10 Potential Issues With This Study

Potential problems exist in this experiment with respect to the implementation of the KK and FR algorithms. The vagueness in the source papers makes it difficult to analyze, criticize [85], and therefore, reproduce the algorithms. To verify the algorithms work as described in the papers, it was necessary to ensure the code could duplicate specific algorithm behaviours as described in the defining figures provided (i.e. Figure 2 in [46], and Figures 8 and 9 in [36]).

Another potential issue with my research is that, although the mental map preservation principles seem related to Gestalt principles in Psychology, both the mental map principles and the Gestalt principles are only descriptive in nature - not explanatory. In other words, the mental map preservation principles can be used to suggest what can be done to preserve a user's mental map, but they may not be able to help explain why.

There may be other measurements that are somehow better related to the mental map preservation layout principles described. However, the general methodology and mental map principles presented may be helpful to graph drawing designers and researchers.

5.2 Helping Designers

System designers create information visualization computer applications which use graph layouts to visualize graph data for users. Algorithm designers create layout adjustment algorithms (like KK and FR) to change graph layouts and help users make decisions [44][2]. Therefore it is important that system and algorithm designers know which algorithm behaviours are going into their systems. Furthermore, the system or algorithm designer may choose to assume what is best for the user or suggest to the user what might work best in certain circumstances (e.g. using particular types of graph data) [63]. Therefore, designers need ways to compare the layout adjustment algorithms within their specific contexts (e.g. user needs, data, application) [68].

5.2.1 The Need For Objective Layout Algorithm Measurements

To help make algorithms iteratively better, measurements and benchmarks are needed to aid comparison of graph layout algorithms [69]. Designers can get this help by using the methodology presented in this study (the layout algorithm comparison 'tools') to make dependable layout algorithm choices.

Both KK and FR are reported to make "nice" and "good" "harmonious" graphs, but, in the literature there is no objective numerical description of the algorithms' specific behaviours with respect to various geometric features that are related to the preservation of the mental map. Himsolt [44] approaches a comparison of algorithms from a conceptually high level that does not get at a statistical characterization of how algorithms behave, and does not relate the findings to a user's mental map. He implemented a number of different algorithms in one computer program (GraphEd) so a graph can be tested with various algorithms. The end result of the research was a subjective ranking of layout criteria [44]. Himsolt goes on to say that balance is often better than optimization; and, displaying the intended structure of a graph is often more important than

formal cost criteria. He believes that doing many trial runs is adequate and is probably the best way to attain information about the practical relevance of layout criteria and algorithms. It follows that the results from a quantitative and objective study performed on a large sample space would give designers confidence that wise algorithm choices are being made. Himsolt's work is definitely a step in the right direction to characterizing layout algorithms, and his claims may be absolutely correct; however, having well accepted principles and objective results that consider a user's mental map seems very important since preserving the mental map during layout changes is crucial for the usability of a system [27][60][34][20][53].

Deciding which algorithm is best depends on the context of the designer's and user's specific needs. Regardless of the exact context of the layout algorithm's use, the designer needs to be able to objectively test layout algorithms so he can make an informed decision of which algorithm is best at preserving the users' mental map [14].

5.2.2 Design Choices Affect Maintenance of the Mental Map

A designer's choice of algorithm may affect a user's ability to maintain their mental map [32]. However, it is difficult to say exactly what the most important behaviour features of a graph layout algorithm are. For example, should an algorithm designer choose to focus on making the code fast, or on preventing/reducing the number of link crossings, or should he strive for a balance between these criteria (and others)?

Inherent tradeoffs occur when layout algorithms try to optimize various layout criteria [33][66], and tradeoffs exist between the mental map preservation principles mentioned in this research. For instance, the preservation of node positions to reduce displacement may be at odds with node movement that would result in fewer link crossings or higher crossing angles [44]. It may be necessary to allow nodes to move along non-optimal paths to avoid link crossings; however, under certain circumstances, it may be preferable to allow some link crossings as the layout is changed instead of making the user watch the layout untangle in a confusing way [34].

This thesis does not attempt to show what measurement or layout attribute is most important to the maintenance of a user's mental map. If researchers can exploit visual mechanisms in the brain, and recognize their limitations [59], graph drawing can better facilitate the preservation of

the mental map [34]. The choice of 'best' auto-layout mechanism must be left to the discretion of the designer, and this important decision may change depending on the rapidly changing state or needs of the particular system/user (e.g. disaster response system in which time is an important factor) [68].

5.2.3 Using Statistical Analysis To Inform Design Choices

Using statistics to compare algorithm behaviour can help designers make informed choices. When studying an algorithm, the goal is insight, not numbers [42]; but, numbers can describe behaviours of algorithms, and thereby the numbers can lead to insight. Instead of ad hoc algorithm development as mentioned by [81], this insight may allow for structured algorithm and system development as part of a comprehensive assessment process that should result in higher quality visualizations [68]. Therefore, although my study uses general graphs and tests only two layout algorithms, the measurements, resulting statistics, and experimental methodology may still be useful to algorithm and system designers.

Statistical analysis can show trends in the data and the trends can be used to predict how an algorithm generally behaves. Statistics can help verify if persistent differences or similarities exist in the two sets of layout data (e.g. comparing results at graph size 15 and 30). A statistical characterization can numerically represent the behaviour of a graph layout algorithm with respect to the chosen measurements.

5.2.4 This Methodology Might Have Been Helpful in Other Studies

The measurements, statistics, and methodology defined in my research might have been helpful in a number of previous studies. In some studies, new FD layout adjustment algorithms were created based on existing FD algorithms (e.g. [33][6][24][8][75][34][85][50][45][55][38]) whereas some studies investigated the use of existing layout algorithms for particular layout purposes [8][79][6][82][29][17][32]. These studies might have benefitted from the methodology presented in my thesis to objectively measure and compare the behaviour of algorithms.

5.2.5 This Study May Not Be Helpful in Certain Circumstances

This work presented in this thesis may not be helpful to designers in certain graph drawing circumstances. For example, if a designer wants particular functionality and is willing to slightly or entirely abandon the mental map principles in order to attain certain layout criteria then this method would not be useful. Depending on the designer's needs, it may be advantageous to allow more node movement or more change from initial to final layout as long as the end result is 'better.' For example, Six, Kakoulis, and Tollis [79] developed a layout adjustment algorithm that allowed for more freedom of node movement. This development resulted in layout benefits including more compaction (smaller layout area and shorter link lengths) and reduced number of bends and crossings. According to the mental map preservation principles, it may be that this algorithm does not pay as much attention to the preservation of a user's mental map because nodes are allowed more freedom to move. However, in this case, perhaps it does not matter because the final layout benefits outweigh the usability cost according to the designers' goals.

In addition to situations in which node positions should be maintained or where mental map preservation is not as important as certain layout criteria, my method may not be helpful to designers who do not expect their user will have a mental map established yet. For example, if a user has not had a chance to learn about the relationships or shape of a graph's drawing, the use of a layout adjustment algorithm will not affect the mental map because it was not established yet. In this case, attempts at preservation would be unnecessary and may cause system inefficiency.

With the exception of node activity, the measurements presented in this study are based on differences that occur in a layout from L to L′. For algorithms that are iterative, many cycles of changes occur along the way to finding a layout solution. Therefore, with the exception of node activity, this study may not be useful for examining specific layout circumstances that occur in the steps between L and L′ (however this study could be adapted to study these in between steps). However, by knowing how much change (with respect to a certain measurement of interest) occurs from L to L′, the time required and how many algorithm cycles were required to arrive at L′, a designer can derive a rate of change measure (e.g. change in number of link crossings per cycle). Since preservation of the mental map is related to the minimizing the rate of layout

change [9], this calculation of layout change rate approach may be useful. This rate of change value is not as specific as a measurement that describes how much change happens at what step of the way from L to L′. However, since number of cycles is a hardware independent measure (unlike time), change per cycle values may be a quite useful method to analyze algorithm behaviours (i.e. with respect to any of the measurements described in 3.2.3 to 3.2.8), and these values would be comparable between graph layout studies.

6. Future Work and Conclusion

The choice of which layout algorithm is best for a particular situation is up to the designer [67], however they may not know how an algorithm will reliably behave. In the past, many different layout algorithms have been created that are based on intuitive observations [67][17]. The mental map is an important idea that is related to a user's ongoing understanding of a graph's layout [27][34][32]. More research needs to be done to formulate and validate measurements that are related to the mental map [9]. Even though layout attributes that are related to the mental map are intuitive [27], they can be described using mathematics and statistics. For example, a numerical approach can be used to objectively measure parts of a graph drawing that are related to shape (e.g. orthogonal order of nodes horizontally and vertically).

The research described in this thesis presents principles that are based on preservation of a user's mental map, and these principles form a theoretical basis for related measurements and statistics. The quantitative analysis presented in this study uses statistics to objectively suggest how KK and FR algorithms will characteristically behave under controlled layout conditions for graph layouts with 15 nodes and 30 nodes. The algorithm behaviours of Kamada-Kawai and Fruchterman-Reingold were evaluated using the measurements and statistics from Chapters 3 and 4. The results were presented and discussed in Chapter 5.

The quantitative methodology presented in this research allows for objective comparison between layout algorithms, and the statistical characterization of a layout algorithm is related to the preservation of a hypothetical user's mental map. While there may be some possible issues with this approach (discussed in Chapter 5), the experiment presented in this study should allow a designer to accurately test their algorithms against one another rather than making algorithm choices based on intuition and time-consuming observations. Also, using a statistical characterization approach may allow a designer to quickly test new experimental algorithm behaviours against others while maintaining confidence that their work is still considering their user's mental map. Using the methodology in this study may help a designer understand inherent

design tradeoffs in the algorithms they plan to use [33] and identify areas for more investigation and progress [73][1].

This research may be helpful to researchers and designers in various fields (e.g. Force-Directed layout researchers, cognitive psychologists studying layout adjustment and perception, GUI designers of systems that use automatic layout adjustment algorithms) who intend to evaluate graph layout algorithms [71][72][17]. Their design judgments could be based on the use of the metrics and the methodology I propose in this study - that is based on the maxim that a user's mental map should be preserved [27][60].

6.1 Future Work

In Chapter 5, it was suggested that there may be tradeoffs between the mental map preservation principles (e.g. minimizing node movements versus reducing link crossings). It was also suggested that it is not entirely known what may adversely affect the perception of a graph drawing or the persistent understanding (mental map) of that layout. Therefore, it is not precisely known how to preserve a user's mental map. Future research that looks at how the mental map principles and proposed measurements (presented in Chapter 3) compare would be quite interesting. Also, it would be interesting to know how a tradeoff works over time and/or space. For example, it would be interesting to investigate at what point lowering the number of link crossings should happen in the layout process (how much of the way from L to L′), or if it is just as beneficial (or better) for the user's mental map if the reduction in link crossings happens only near the end of the layout process.

Although cognitive psychology is generally outside the field of graph drawing research, more work is needed in experimental psychology to help understand how the mental map is related to graph layout. Studies have been done to look at how drawing properties are related to graph understanding [71][88], however more research is still needed [68][45]. This area of study seems quite challenging because results and decisions based on those results may depend on the users, the graph data (e.g. number of node, number of links, and other factors), and the graph drawing context (e.g. the user's task, visualization area, size of each node, type of interface) [2]. It seems like a very difficult challenge to design and perform a well controlled user study with adequate

sample size because many sources of variability exist (and may be confounding). Future research into the psychology of how a user deals with a graph drawing would help researchers and designers understand which layout qualities are most important to help a user better use graph drawings [71]. The various layout criteria presented in this study are all important, but their effects are hard to separate to determine the relative importance of each to mental map preservation.

Once there is more cognitive research done in the field of preserving a user's understanding of a graph layout, it may be helpful and interesting to compare how combinations of layout criteria might be complementary and/or synergistic. For instance, it may be that minimizing activity and reducing link crossings is far more helpful at maintaining the mental map than reducing displacement and reducing link crossings. Furthermore, the use of measurements to objectively and quantitatively compare layout algorithm behaviours may be one of the first steps to allow for the development of hybrid layout techniques that might be able to generate customized drawings that reflect users' preferences while still adhering to principles of perceptual psychology and/or the mental map [9][53].

In addition to looking at the psychological principles underlying mental map preservation, it would be interesting to examine how the mental map statistics relate to KK and FR when applied to real-world graphs instead of random initial layouts. This next step would lend more credibility to the usefulness of the proposed measurements.

It would also be interesting to conduct this experiment on a set of larger graph sizes. There may be an interesting trend, or perhaps there is some size threshold at which point KK levels out at a particular efficiency (time/cycle). In this experiment, a graph size of 40 nodes was attempted but not used because KK took a very long time to run and it depended heavily on the friction scalar to stop. Perhaps there is another strategy that could help KK find a layout solution more quickly without sacrificing quality with respect to one or more of the statistics presented in Chapter 4.

6.2 Conclusion

There is a need for measurements to compare layout algorithm behaviours that are related to the preservation of a user's mental map [55][9]. My approach was to find well-accepted mental map

preservation principles in the graph drawing literature and develop measurements and statistics that mathematically arise from those principles. Furthermore, I was able to relate these principles to certain Gestalt principles from Psychology. By collecting mental map preservation related principles and developing related measurements and statistics, this study took one of the initial necessary steps towards designers/researchers being able to formally and objectively evaluate and compare graph layout algorithms with respect to maintaining a user's mental map.

The measurements defined in my thesis were used to compare two well-known layout algorithms (KK and FR) in order to validate the idea that the measurements and statistics presented could be used to describe and compare algorithm behaviour with respect to the mental map principles and the preservation of a hypothetical user's mental map. Like Gansner's research [39], my study tried to look at graph layouts generally so the specifics of the data did not get in the way. The measurements and statistics used are simple to calculate and understand, and they are relevant to mental map preservation as discussed in the graph layout adjustment literature. Measurements were computed over many trial runs from initial and final layouts, and statistics were computed.

The results of the experiment were presented and discussed in Chapter 5. With the exception of link length (which was needed as a control variable), the results show that the mental map-related measurements and statistics presented in this study are able to show algorithm behaviour differences that relate to the mental map preservation principles.

The main goal of my research was to gather principles from the graph layout literature that can be used to show the behaviour of force directed algorithms in the context of mental map preservation. The measurements and statistics I have collected and developed were tested on two common force directed algorithms, and the results give a clear quantitative picture of how the algorithms act. My study shows that the measurements presented can be used to quantitatively demonstrate that, FR generally outperforms KK. Furthermore, in terms of the supporting principles, it can be safely said that FR is probably better than KK at preserving a user's mental map for the graph sizes tested.

Reference List

[1] Anandan, P., "A Computational Framework and an Algorithm for the Measurement of Visual Motion," *International Journal of Computer Vision*, vol. 2, no. 3, pp. 283-310, 1989.

[2] Arunachalam, V., Pei, B. K. W., and Steinbart, P. J., "Impression management with graphs: effects on choices," *Journal of Information Systems*, vol. 16, no. 2, pp. 183-202, 2002.

[3] Barnard, S. T. and Thompson, W. B., "Disparity analysis of images," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 2, no. 4, pp. 333-340, 1980.

[4] Beard, M. K., Buttenfield, B. P., and Clapham, S. B., "NCGIA Research Initiative 7: Visualization of Spatial Data Quality," *National Center for Geographic Information and Analysis: Technical paper 91-26*, pp. 1-175, 1991.

[5] Bergman, D. S., Youssef, B. B., Bizzocchi, J., and Bowes, J., "Interpolation techniques for the artificial construction of video slow motion in the postproduction process," *Third International Conference on Advances in Computer Entertainment*, Hollywood, CA, vol. 66, June 2006.

[6] Bertault, F., "A force-directed algorithm that preserves edge-crossing properties," *Information Processing Letters*, vol. 74, no.1-2, pp. 7-13, 2000.

[7] Brandenburg, F. J., Himsolt, M., and Rohrer, C., "An Experimental Comparison of Force-Directed and Randomized Graph Drawing Algorithms," *Proceedings Graph Drawing*, pp. 76-87, 1996.

[8] Brandes, U., and Wagner, D., "A Bayesian Paradigm for Dynamic Graph Layout," *Proceedings Graph Drawing*, pp. 236-247, 1997.

[9] Bridgeman, S., and Tamassia, R., "Difference Metrics for Interactive Orthogonal Graph Drawing Algorithms," *Journal of Graph Algorithms and Applications*, vol. 4, no. 3, pp. 47-74, 2000.

[10] Brosnan, M. J., Scott, F. J., Fox, S., and Pye, J., "Gestalt processing in autism: failure to process perceptual relationships and the implications for contextual understanding," *Journal of Child Psychology and Psychiatry*, vol. 45, no. 3, pp. 459-469, 2004.

[11] Burt, P., "Multiresolution Techniques for Image Representation, Analysis, and 'Smart' Transmission," *Proceedings of SPIE Conference*, vol. 1199, pp. 2-15. Nov. 1989.

[12] Calamoneri, T., Jannelli S., and Petreschi, R., "Experimental comparison of graph drawing algorithms for cubic graphs," *Journal of Graph Algorithms and Applications*, vol. 3, no. 2, pp. 1-23, 2000.

[13] Chang, D., Nesbitt, K. V., and Wilkins, K., "The gestalt principles of similarity and proximity apply to both the haptic and visual grouping of elements," *Proceedings of the Eight Australasian Conference on User interface*, Ballarat, Victoria, Australia, vol. 64, pp. 79-86. Jan.-Feb. 2007.

[14] Chen, C., *Information Visualisation and Virtual Environments*, Springer-Verlag, 1999.

[15] Chupeau, B., and Francois, E., "Region-based Motion Estimation for Content-based Video Coding and Indexing," *SPIE Visual Communications and Image Processing*, vol. 4067, pp. 884-893, 2000.

[16] Cohen, R. F., Di Battista, G., Tamassia, R., Tollis, I. G., and Bertolazzi, P., "A framework for dynamic graph drawing," *Proceedings of the Eighth Annual Symposium on Computational Geometry*, New York, NY: ACM Press, pp. 261-270, 1992.

[17] Cui, W., Zhou, H., Qu, H., Wong, P.C., and Li, X., "Geometry-Based Edge Clustering for Graph Visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1277-1284, 2008.

[18] Davidson, R., and Harel, D., "Drawing graphs nicely using simulated annealing," *ACM Transactions on Graphics*, vol. 15, no. 4, pp. 301-331, 1996.

[19] Deregowski, J.B., "Symmetry, gestalt and information theory," *The Quarterly Journal of Experimental Psychology*, vol. 23, no. 4, pp. 381-385, 1971.

[20] Di Battista, G., Eades, P., Tamassia, R., and Tollis, I., *Graph drawing: algorithms for the visualization of graphs*, Prentice-Hall, 1999.

[21] Diaz, J., Petit, J., and Serna, M., "A survey of graph layout problems," *ACM Computing Surveys*, vol. 34, no. 3, pp. 313-356, Sept. 2002.

[22] Dornheim, C., "Planar Graphs with Topological Constraints," *Journal of Graph Algorithms and Applications*, vol. 6, no. 1, pp. 27-66, 2002.

[23] Dustin, E., *Effective Software Testing: 50 Specific Ways To Improve Your Testing*. Toronto: Addison-Wesley, 2003.

[24] Dwyer, T., Marriot, K., Stuckey, P. J., "Fast node overlap removal," *Lecture Notes in Computer Science*, vol. 3843, pp. 153-164, 2005.

[25] Eades, P., and de Mendonca, C. F. X., "Vertex splitting & tension-free layout," *Graph Drawing*, pp. 202-211, 1995.

[26] Eades, P., Lin, X., "Spring Algorithms and Symmetry," *Theoretical Computer Science*, vol. 240, no. 2, pp. 379-405, 1999.

[27] Eades, P., Lai, W., Misue, K., and Sugiyama, K., "Preserving the mental map of a diagram," *Proceedings of Compugraphics*, vol. 91, pp. 24-33, 1991.

[28] Eades, P., "A heuristic for graph drawing," *Congressus Numerantium*, vol. 42, pp. 149–60, 1984.

[29] Eisenmann, H., and Johannes, F. M., "Generic global placement and floorplanning," *Proceedings of the 35th Annual Conference on Design Automation*, New York: ACM Press, pp. 269-274. June 1998.

[30] Eschbach, T., Günther, W., Drechsler, R., and Becker, B., "Crossing Reduction by Windows Optimization," *Tenth International Symposium on Graph Drawing*, Irvine, CA, pp. 285-294, Aug. 2002.

[31] Forster, M., and Bachmaier, C., "Clustered Level Planarity," *Lecture Notes in Computer Science*, vol. 2932, pp. 218–228, 2004.

[32] Freire, M., and Rodríguez, P., "Preserving the mental map in interactive graph interfaces," *Proceedings of the Working Conference on Advanced Visual interfaces*, Venezia, Italy, pp.270-273, May 2006.

[33] Frick, A., Ludwig, A., and Mehldau, H., "A Fast Adaptive Layout Algorithm for Undirected Graphs (Extended Abstract and System Demonstration)," *Proceedings Graph Drawing*, pp. 388-403, 1994.

[34] Friedrich, C., and Eades, P., "Graph Drawing in Motion," *Journal of Graph Algorithms and Applications*, vol. 6, no. 3, pp. 353-370, 2002.

[35] Frishman, Y., and Tal, A., "Online Dynamic Graph Drawing," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 4, pp. 727-740, July 2008.

[36] Fruchterman, T., and Reingold, E., "Graph drawing by force-directed placement," *Software-Practice Experience*, vol. 21, pp. 1129–1164, 1991.

[37] Gajer, P., Goodrich, M. T., and Kobourov, S. G., "A multi-dimensional approach to force-directed layouts of large graphs," *Computational Geometry: Theory and Applications*. vol. 29, no. 1, pp. 3-18, 2004.

[38] Gansner, E. R., Mocenigo, J. M., and North, S. C., "Visualizing software for telecommunication services," *Proceedings of the 2003 ACM Symposium on Software Visualization*, San Diego, CA, pp. 151-ff, June 2003.

[39] Gansner, E. R., and North, S., "Improved Force-Directed Layouts," *Proceedings Graph Drawing*, pp. 364-373, 1998.

[40] Garey, M., and Johnson, D., “Crossing Number is NP-Complete,” *SIAM Journal of Algebraic and Discrete Methods*, vol. 4, no. 3, pp. 312-316, 1983.

[41] Gelman, A., Pasarica, C., and Dodhia, R., "Let's Practice What We Preach: Turning Tables Into Graphs in Statistics Research," *The American Statistician*, vol. 56, no. 2, pp. 121-130, May 2002.

[42] Groller, E., “Insight into Data through Visualization,” *Lecture Notes In Computer Science*, vol. 2265, pp. 352-355, 2001.

[43] Herman, I., Melançon, G., and Marshall, M. S., “Graph Visualization and Navigation in Information Visualization: A Survey,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 6, no. 1, pp. 24-43, 2000.

[44] Himsolt, M., “Comparing and Evaluating Layout Algorithms within GraphEd,” *Journal of Visual Languages and Computing*, vol. 6, no. 3, pp. 255-273, 1995.

[45] Huang, X., Eades, P., and Lai, W., “A framework of filtering, clustering and dynamic layout graphs for visualization,” *Proceedings of the Twenty-Eighth Australasian Conference on Computer Science*, Newcastle, Australia, vol.38, pp. 87-96, 2005.

[46] Kamada, T. and Kawai, S., “An algorithm for drawing general undirected graphs,” *Information Processing Letters*, vol. 31, no. 1, pp. 7-15. April 1989.

[47] Kaufmann, M., and Wagner, D., *Drawing Graphs, Methods and Models*, Springer, 2001.

[48] Klemetti, H., Lapinleimu, I., Mäkinen, E., and Sieranta, M., “A programming project: trimming the spring algorithm for drawing hypergraphs,” *SIGCSE Bulletin*, vol. 27, no. 3, pp. 34-38, 1995.

[49] Koningsberg, I., *The Complete Film Dictionary*. 2nd ed. London: Bloomsbury, 1997.

[50] Koren, Y., and Harel, D., “Axis-by-Axis Stress Minimization,” *Proceedings Graph Drawing*, pp. 450-459, 2004.

[51] Kyrylov, V., Bergman, D. S., and Greber, M., “Multi-criteria optimization of ball passing in simulated soccer,” *Journal of Multi-Criteria Decision Analysis*, vol. 13, no. 2-3, pp. 103-113, 2006.

[52] Le Gall, D., “MPEG: a video compression standard for multimedia applications,” *Communications of the ACM*, vol. 34, no. 4, pp. 46-58, 1991.

[53] Lee, Y., Lin, C., and Yen, H., “Mental map preserving graph drawing using simulated annealing,” *Proceedings of the 2006 Asia-Pacific Symposium on information Visualisation*, Tokyo, Japan, vol. 60, pp. 179-188, 2006.

[54] Lehar, S., "Gestalt Isomorphism and the Primacy of Subjective Conscious Experience: A Gestalt Bubble Model," *Behavioral & Brain Sciences*, vol. 26, no. 4, pp. 375-444, 2004.

[55] Li, W., Eades, P., and Nikolov, N., "Using spring algorithms to remove node overlapping," *Proceedings of the 2005 Asia-Pacific Symposium on information Visualisation*, Sydney, Australia, vol. 45 pp. 131-140, 2005.

[56] Lyons, K. A., Meijer, H., and Rappaport, D., "Algorithms for Cluster Busting in Anchored Graph Drawing," *Journal of Graph Algorithms and Applications*, vol. 2, no. 1, pp. 1-14, 1998.

[57] Marriott, K., Stuckey, P., Tam, V., and He, W., "Removing Node Overlapping in Graph Layout Using Constrained Optimization," *Constraints*, vol. 8, pp. 143-171, 2003.

[58] Merrick, D., and Gudmundsson, J., "Increasing the readability of graph drawings with centrality-based scaling," *Proceedings of the Asia Pacific Symposium on information Visualisation*, Tokyo, Japan, vol. 60, pp. 67-76, Feb. 2006.

[59] Miller, G. A., "The magical number seven, plus or minus two: some limits on our capacity for processing information," *Psychological Review*, vol. 63, pp. 81—97, 1967.

[60] Misue, K., Eades, P., Lai, W., and Sugiyama, K., "Layout Adjustment and the Mental Map," *Journal of Visual Languages and Computing*, vol. 6, no. 2, pp. 183-210, 1995.

[61] Mullet, K., and Sano, D., *Designing Visual Interfaces: Communication Oriented Techniques*. Englewood Cliffs, NJ: Prentice-Hall. 1995.

[62] Newbery, F. J., "Edge concentration: a method for clustering directed graphs," *SIGSOFT Software Engineering Notes*, vol. 14, no. 7, pp. 76-85, 1989.

[63] Niggemann, O., and Stein, B., "A meta heuristic for graph drawing: learning the optimal graph-drawing method for clustered graphs," *Proceedings of the Working Conference on Advanced Visual interfaces*, Palermo, Italy, pp. 286-289, 2000.

[64] Noack, A. and Lewerentz, C., "A space of layout styles for hierarchical graph models of software systems," *Proceedings of the 2005 ACM Symposium on Software Visualization*, St. Louis, Missouri, pp. 155-164, May 2005.

[65] North, S. C., "Incremental layout in DynaDAG," *Lecture Notes in Computer Science*, vol. 1027, pp. 409-418, 1996.

[66] North, S. C., *Drawing graphs with NEATO (user manual)*, 2004.

[67] Pajntar, B., "Overview of algorithms for graph drawing," *The Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Philadelphia, USA, pp. 21-25, Aug. 2006.

[68] Pfitzner, D., Hobbs, V., and Powers, D., "A unified taxonomic framework for information visualization," *Proceedings of the Asia-Pacific Symposium on information Visualisation*, Adelaide, Australia, vol. 24, pp. 57-66, 2003.

[69] Plaisant, C., "The challenge of information visualization evaluation," *Proceedings of the Working Conference on Advanced Visual interfaces*, Gallipoli, Italy, pp 109-116, May 2004.

[70] Purchase, H. C., Carrington, D., and Allder, J., "Empirical Evaluation of Aesthetics-based Graph Layout," *Empirical Software Engineering*, vol. 7, no. 3 pp. 233-255, Sep. 2002.

[71] Purchase, H. C., Cohen, R. F., and James, M. I., "An experimental study of the basis for graph drawing algorithms," *Journal of Experimental Algorithmics*, vol. 2, article. 4, Jan. 1997.

[72] Purchase, H. C., McGill, M., Colpoys, L., and Carrington, D., "Graph drawing aesthetics and the comprehension of UML class diagrams: an empirical study," *Proceedings of the 2001 Asia-Pacific Symposium on information Visualisation*, Sydney, Australia, vol. 9, pp. 129-137, 2001.

[73] Ramos, G., Robertson, G., Czerwinski, M., Tan, D., Baudisch, P., Hinckley, K., and Agrawala, M., "Tumble! Splat! helping users access and manipulate occluded content in 2D drawings," *Proceedings of the Working Conference on Advanced Visual interfaces*, Venezia, Italy, pp. 428-435, May 2006.

[74] Rothrock, L., Barron, K., Simpson, T. W., Frecker, M., Ligetti, C., and Barton, R., "Applying the Proximity Compatibility and the Control-Display Compatibility Principles to Engineering Design Interfaces," *Human Factors and Ergonomics in Manufacturing*, vol. 16, no. 1, pp. 61-81, 2006.

[75] Ryall, K., Marks, J., and Shieber, S., "An Interactive System for Drawing Graphs," *Proceedings Graph Drawing*, pp. 387-394, 1997.

[76] Sadahiro, Y., "Cluster perception in the distribution of point objects," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 34, no. 1, pp- 49-62, 1997.

[77] Salh, B., Bergman, D., Marotta, A., and Pelech, S.L., "Differential cyclin-dependent kinase expression and activation in human colon cancer," *Anticancer Research,* vol. 19, no. 1B, pp.741-748, Jan-Feb. 1999.

[78] Sears, A., "AIDE: a step toward metric-based interface development tools," *Proceedings of the 8th Annual ACM Symposium on User interface and Software Technology*, Pittsburgh, PA, pp. 101-110, Nov. 1995.

[79] Six, J. M., Kakoulis, K. G., and Tollis, I. G., "Techniques for the Refinement of Orthogonal Graph Drawings," *Journal of Graph Algorithms and Applications*, vol. 4, no. 3, pp. 75-103, 2000.

[80] Steinman, R. M., Pizlo, Z., Pizlo, F. J., "Phi is not beta, and why Wertheimer's discovery launched the Gestalt revolution," *Vision Research*, vol. 40, no. 17, pp. 2257-2264, 2000.

[81] Storey, M. D., "A cognitive framework for describing and evaluating software exploration tools," PhD Thesis, Simon Fraser University, 1998.

[82] Storey, M. D., and Müller, H. A., "Graph Layout Adjustment Strategies," *Lecture Notes In Computer Science*, vol. 1027, pp. 487-499, 1995.

[83] Tandjung, S. S., Gunawan, T. S., and Nang, C. M., "Motion estimation using adaptive matching and multiscale methods," *International Conference On Visual Communications and Image Processing*, Perth, Australia, vol. 4067, pp. 1344-1355, June 2000.

[84] Tekalp, A. M., *Digital Video Processing*. New Jersey: Prentice Hall Inc. 1995.

[85] Tunkelang, D., "A Numerical Approach to General Graph Drawing," PhD Thesis, Carnegie Mellon University, 1999.

[86] van Ham, F., and Rogowitz, B., "Perceptual Organization in User-Generated Graph Layouts," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1333-1339, Nov. 2008.

[87] Ware, C., and Bobrow, R., "Motion to support rapid interactive queries on node-link diagrams," *ACM Transactions on Applied Perception*, vol. 1, no. 1, pp. 3-18, 2004.

[88] Ware, C., Purchase, H., Colpoys, L., and McGill, M., "Cognitive measurements of graph aesthetics," *Information Visualization*, vol. 1, no. 2, pp. 103-110, 2002.

[89] Watts, D. J., and Strogatz, S. H., "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, pp. 440–442, 1998.

[90] Wertheimer, M., "Experimental Studies of the Perception of Movement," *Zeitschrift fur Psychologie,* vol. 61, pp. 161-265, 1912.

[91] Wickens, C. D., Lee, J. D., Liu, Y., and Gordon-Becker, S., *An Introduction to Human Factors Engineering*, 2nd ed. New Jersey: Pearson Prentice Hall, 2004.

Appendices

Appendix 1: Calculations for DDLOO and GCOO

Below are the DDLOO and GCOO calculations for the layout changes shown in Figure 3.2.5.1.

DDLOO calculations

The DDLOO calculations consider vertical changes first, horizontal changes second, and their sum is the DDLOO value for a layout.

The DDLOO vertical calculations are:

- all the nodes directly-linked to n1 do not move so no orthogonal ordering change is recorded for any of them
- all the nodes from n1 along the linked chain of nodes out to n10 also do not move, they maintain their relative positions, and no change is recorded for these nodes either
- node n3 has 3 directly-linked nodes (n2, n4, and n5), it starts with its directly-linked neighbors above, and ends up with n5 below (+1)
- n4 stays above n3 (which is its only directly-linked neighbor) so there is no orthogonal ordering change recorded for this node
- n5 goes below n3 (+1) and ends up above n6 (+1)
- n6 ends up below n5 which is its only directly-linked neighbor (+1).

The DDLOO horizontal calculations are:

- node n3 maintains the relative horizontal positions for all its directly-linked neighbors, so no horizontal change is registered for this node
- n4 stays right of its only directly-linked neighbor n3, so no change is registered for this node
- n5 stays right of n3 and left of n6, so no change is registered for this node

- n6 stays to the right of n5 so there is no change
- all other nodes are not moved, so there is no change registered for them.

The DDLOO value is obtained by adding the horizontal and vertical DDLOO values, which comes to 4 (4+0). Note that some relative positional changes are recorded twice. This is done intentionally because eventually the DDLOO is averaged for all nodes in the layout.

GCOO calculations

The GCOO calculations consider vertical changes first, followed by horizontal changes, and their sum is the GCOO value for a layout.

The GCOO vertical calculations are:

- node n1starts with 9 nodes above and ends with 6, so the absolute value of the change is (|6 - 9| = +3)
- n2 has a global orthogonal order change of (|7 - 10| = +3)
- n3 has two nodes go below it (|10 - 12| = +2)
- n4's change is (|7 - 0| = +7)
- n5's change is (|11 - 2| = +9)
- n6's change is (|12 - 1| = +11)
- n7's change is (+3)
- n8's change is (+3)
- n9's change is (+3)
- n10's change is (+3)
- n11's change is (+2)
- n12's change is (+3)
- n13's change is (+3)

The GCOO horizontal calculations are:

- all the nodes to the left of n10 do not have any horizontal orthogonal order changes because no nodes move past n10 to the left, therefore the only nodes to consider are n4, n5, n6, and n10
- n4's change is (+1)

- n5’s change is (+3)
- n6 stays the furthest right, so it has no horizontal orthogonal ordering change
- n10’s change is (+1)

The GCOO value is obtained by adding the horizontal and vertical GCOO values, which is (44 + 5 = 49).