Graph Interaction through Force-Based Skeletal Animation

Colin Murray **Damian Merrick** **Masahiro Takatsuka**

School of Information Technologies,
The University of Sydney,
Australia,
Email: {cmurray,dmerrick,masa}@it.usyd.edu.au

Abstract

Skeletal animation is a concept that has been used in the areas of motion pictures and computer games to create realistic motion for the animation of articulated characters. Recent work has applied skeletal animation techniques from inverse kinematics to the field of graph interaction. The previous work introduced an interesting idea suggesting a skeletal graph interaction system would be intuitive and natural. However, due to problems with the skeletal animation techniques used, the benefits of the system were limited. This paper investigates the use of a new dynamics based technique previously used in the area of skeletal animation for graph interaction. The motivation for this work was to improve upon the previous work by providing a more intuitive skeletal graph interaction system. An intuitive skeletal interaction system could reduce the difficulty and time taken in navigating the graph and increase the amount of information that can be interpreted and understood. This technique has been implemented in a graph interaction environment in Java3D allowing the user to drag nodes in order to manipulate the layout of the graph. The graph reacts to the user movement while maintaining the physical constraints placed upon it by the skeletal metaphor.
Keywords: Graph Drawing, Graph Interaction, Skeletal Animation

1 Introduction

A relational network or graph is commonly used to represent relations between objects. A graph consists of a set of nodes and edges where nodes represent objects and edges represent the relationship between objects.

Graph drawing takes the set of nodes and edges of a graph and assigns coordinates to the nodes so that they can be drawn. This produces a drawing that provides a geometric representation for a set of relational data. The aim of graph drawing is to produce a picture that allows the viewer to easily understand the information that is represented by a graph. A lot of research has been done in the area of graph drawing and this is given in more detail in the following section.

While a number of good graph drawing algorithms have been developed, there are always situations, particularly in 3D, where the layout provided is difficult to understand. In these situations it is desirable to have a mechanism to allow the viewer to modify both the graph layout and their view of this layout. Graph interaction techniques are designed to provide this mechanism.

Graph interaction allows the user to make changes to a graph drawing. The interaction technique used determines how the graph will respond to input from the user. Graph interaction is a field that aims to complement graph drawing by reducing the difficulty and time taken in understanding the information that is being represented by a graph. Effective interaction techniques have the potential to reduce the time that an analyst needs in order to navigate the presented information, and increase the amount of information that can be interpreted and understood.

One such way in which we can control how movements by the user will result in changes to the graph is through the use of a physical metaphor. The aim of a physical metaphor is to simulate a physical structure that the user is familiar with. This will result in changes that feel natural to the user. Any graph interaction technique should feel natural and intuitive, and a useful physical metaphor is able to provide this.

A physical metaphor that was recently introduced to graph interaction (Merrick 2002) is the skeletal structure. A skeleton in this case may be seen as a system of bones of fixed length connected by joints that may be rotated at any angle.

Using the skeletal structure for graph interaction has many advantages. Firstly, it is easily mapped onto a graph. The nodes of the graph are mapped to the joints of the skeleton and the edges of the graph are mapped to the bones of the skeleton. As the user drags a node, other nodes will move and joints will rotate. The interaction technique must ensure that edge-lengths remain constant.

The second major advantage of the skeletal metaphor is that a large amount of research has been performed in the area of skeletal animation. The majority of skeletal animation research has been driven by its need in robotics, computer games and motion pictures. These algorithms can also be applied to the graph structure. As a result we have a large number of algorithms to choose from and are likely to be able to find a technique that is suited to both skeletal animation and graph interaction.

The skeleton is also a structure that is manipu-

lated by humans on a regular basis and therefore, it should feel natural and intuitive.

The aim of this paper is to investigate graph interaction methods and the use of a graph interaction system that allows the analyst to understand a graph's structure more quickly and easily than other graph interaction systems. This goal is achieved by using a skeleton as a physical metaphor for a graph and using a skeletal animation technique that is very highly suited for this purpose.

2 Background

Graphs have been used for a long time in conveying information and as a result of this a lot of research has been done in the area of graph drawing. Di Battista et al. provide a good review of this research (Di Battista et al. 1994, Di Battista et al. 1999).

Various aesthetic criteria have been discussed which attempt to measure the readability of a graph drawing by general optimisation goals (Esposito 1988). Examples of aesthetic criteria include minimisation of crossings, area, bends and maximisation of smallest angle and symmetries (Di Battista et al. 1994). In many cases, it is not possible to completely satisfy more than one aesthetic criteria simultaneously. It is also common to have an additional criterion requiring that certain constraints are satisfied (Dengler et al. 1993). Examples of possible constraints are keeping a certain vertex in a fixed position or keeping a group of vertices close together. An algorithm for producing a graph drawing is known as a layout algorithm. As a result of the large amount of research into graph drawing many layout algorithms have been developed that attempt to optimise specific aesthetic criteria.

The focus in this paper is on general undirected graphs drawn in three dimensions for which force-directed methods are often used. Force-directed methods define a system of forces acting on the vertices and edges. The graph drawing then evolves to a minimum energy state. Force-directed methods are relatively easy to implement, can easily be extended to three dimensions and the smooth evolution of the drawing helps to preserve the user's mental map as the layout changes.

Force-directed methods are widely used in the graph drawing field and many different force-directed algorithms exist. The spring embedder method (Eades 1984) represents edges as springs with unit natural length and non-connected vertices in the graph are connected by springs in the system with infinite natural length. As a result connected vertices are attracted to each other unless they are closer than unit length together. In that case they will repel each other. Non-connected vertices will repel each other. The system is then let go until a minimum energy state is reached.

Evaluation of the benefits provided by the aesthetic criteria on which graph drawing algorithms are based has produced mixed results (Batini et al. 1985, Ding & Mateti 1990, Purchase et al. 1997, Purchase et al. 2001). In any automated graph drawing system there will be situations where the initial graph drawing provided is not easy to understand. Also, many automated graph drawing systems provide an analyst with only one drawing of a graph. If the viewer is unable to understand the graph based on this one drawing, they will have no further avenue in which to comprehend the information. By providing the user with a mechanism for changing the layout of the graph we facilitate a potential increase in the users understanding of the graph. Studies into the effectiveness of graph interaction techniques support this (Herman et al. 2000).

Graph interaction techniques determine how the graph reacts to actions by the user. The main problem in graph interaction is providing a mechanism that will ensure that the graph becomes and remains understandable as the user navigates the graph. It is desirable that the graph will react in a way that is natural and intuitive (Merrick 2002). Graph interaction techniques attempt to provide solutions for this problem.

A number of techniques have been used to aid the interaction of a user with a graph. Techniques that are used to control the view of a fixed graph drawing include the operations of rotation, scaling and translation. In this paper we will be focusing on methods that allow the user control over the positions of the nodes themselves. These methods give the user greater control over how the graph is drawn as the embedding of the graph can also be changed. A particular embedding of a graph drawing defines the ordering of edges around each node. While a given graph has an infinite number of possible graph drawings it has a possibly very large but finite number of embeddings.

While we want to give the user a degree of freedom in navigating the graph, we also want to maintain some control over the layout of the graph to ensure certain properties remain unchanged. Placing such constraints over the control of the graph will ensure that we continually provide an aesthetically pleasing graph drawing and this will assist the user in finding an understandable layout. In addition it may reduce the time taken for the user to achieve such a layout.

When a graph is drawn in two dimensions it is easier for a user to navigate the graph. In terms of changing the view of the graph, interaction in two dimensions is usually limited to scrolling or zooming and this is only necessary for graphs that exceed the size of the screen. It is the positions of the nodes and the way in which the edges are drawn that will determine how easily the graph can be understood, as these factors alone will determine the aesthetic properties of the graph. This is because there is no depth to the drawing. As a result, it is methods that allow the user to change the graph layout that are most important for two dimensional graph interaction. Mechanisms that allow layout changing modifications are usually only necessary if the drawing provided by the original layout algorithm is incomprehensible. There will always be some situations where this is the case, and therefore such mechanisms are important.

When a graph is drawn in three dimensions, navigating the graph is much more difficult. On regular displays the user is only seeing a two dimensional projection of a three dimensional image. In any given view, it is likely that part of the graph will be obscured due to the additional depth of the drawing and lack of depth of the display device. The user will need to be able to change their view of the graph in order to experience all three dimensions. In any number of dimensions it is good to start with a good graph drawing. However, in three dimensions, in

addition to the positions of the nodes, an important aspect that will determine how easily a user can understand the graph structure is their ability to modify their view of the graph. Graph Interaction in three dimensions is even more important than in two dimensions as there are many additional factors that could lead to an original graph drawing being incomprehensible. This also makes graph interaction in three dimensions a more difficult problem. Three dimensional graph drawing mechanisms need to provide the ability to change both the layout and view of a graph. Also, these mechanisms are more complicated in three dimensions. There are many more ways in which the layout can be changed as node movement isn't restricted to a single plane, and when changing the view, rotation needs to be provided in addition to translation and zooming.

Not only is graph interaction in three dimensions more difficult, it is also more necessary. As drawing graphs in 3D is more complicated, there are likely to be even more situations where the initial graph drawing is difficult to understand.

Graph interaction is a field that has become more prevalent due to a more widespread use of three dimensions in graph drawing. There are a number of reasons why this is the case. Usability studies (Ware et al. 1993, Ware & Franck 1994, Ware & Franck 1996, Cockburn & McKenzie 2001) have shown the benefits of three dimensional drawings over two dimensional drawings. These studies have shown that three dimensional drawings are more easily understood, that it is easier to follow the structure of the graph and that it feels more realistic and natural. A wider availability of hardware capable of drawing graphs in three dimensions has also lead to three dimensional graph drawings becoming more widespread.

While there have been studies showing the benefits of three dimensional drawings, there are also studies claiming that it is better to draw graphs in two dimensions (Nielsen 1998). Cases where three dimensions have been shown to be not useful may be due to a poor interaction model, as a good interaction model would have improved the understandability of the graph. There are certainly cases where three dimensional graph drawings are useful and they can only become more useful with a good interaction model (Cockburn & McKenzie 2001). A good interaction model will ensure that the benefits of three dimensions are fully realised. While graph drawing is a field that has been thoroughly explored (Di Battista et al. 1994, Di Battista et al. 1999), graph interaction remains a relatively new field and novel methods that aid the user's understanding of the graph structure, particularly in three dimensions, are still required.

While navigating the graph visualisation space, the user builds a mental map (Gould & White 1986, Eades 1991, Misue et al. 1995) of the structure of the graph. Being able to maintain and add to this mental map as they navigate the graph is essential for developing an understanding of the structure of the graph. The graph interaction mechanism must not make drastic, unintuitive changes to the drawing of the graph. Graph animation (Friedrich 2002) is a method that demonstrates the transition between drawings of a graph. An intuitive graph interaction mechanism, with the aid of animation allows the user to easily follow the transition from the initial graph drawing to the final graph drawing. This ensures that the mental map is maintained.

The simplest way of providing graph interaction is to use an adaptive graph drawing algorithm. An adaptive graph drawing algorithm produces a drawing that is dependent upon the initial geometric properties given to the graph. These types of algorithms effectively take a graph drawing and their aim is to improve upon it. Adaptive graph drawing algorithms are useful for graph interaction tasks as they are more likely to preserve the user's mental map. The speed requirements of an algorithm used for graph interaction is higher than that of general graph drawing. Algorithms developed with graph interaction in mind (Bruβ & Frick 1996) are designed to be very fast.

One method used in graph interaction involves placing a physical metaphor on the structure of the graph such that the graph responds to movements of nodes in a way corresponding to the physical metaphor. A common physical metaphor in undirected graph drawing is the force-directed paradigm and this idea is easily extended to graph interaction as force-directed algorithms are adaptive. Forces are recalculated as the user moves a node and the position of the nodes are updated accordingly.

Skeletal animation is a concept that is commonly used in the areas of computer games and motion pictures. As a result a large number of techniques have been developed that control the movement of a skeleton in a way that is designed to appear natural to the viewer.

Recent work has applied the skeleton as a physical metaphor to the field of graph interaction (Merrick 2002). By applying skeletal constraints on a graph we can provide a physical metaphor that feels natural to the user and the time taken in navigating the graph is reduced.

The focus of the previous work was on inverse kinematic techniques. Inverse kinematic methods move a joint to a desired position while controlling the movement of several other joints and taking into account positions of fixed joints, bone lengths and joint angles. These techniques are often used in the area of robotics.

While inverse kinematics techniques operate on the geometry of articulated structures, with dynamics, all interactions with the system are modelled as forces acting upon rigid and articulated bodies. Inverse kinematics techniques have been useful for robotics, and as a result, it has been the topic of the majority of research in skeletal animation. However, there has also been some development of dynamics-based techniques, a lot of which has been driven by the need for skeletal animation techniques in computer games, where the main goals are believability and speed of execution. A graph interaction environment shares these goals. Therefore, dynamics-based techniques may be more suited for graph interaction purposes. This supports investigation into dynamics-based skeletal animation techniques for graph interaction.

In order to achieve believability, dynamics-based techniques focuses on emulating motion as seen in real environments. These methods are less focused on goal directed motion. These methods use an iterative approach where, at each iteration, the forces upon the particles that make up a body, are summed in order to determine their following positions. This

process is repeated in order to generate motion.

The previous approaches (Merrick 2002) used to implement the skeletal metaphor in a graph interaction system provided a natural way for the user to navigate a graph. However, these methods could be improved upon in terms of the satisfaction of the constraints defining the skeleton, the naturalness of the motion and the ability to preserve a user's mental map between successive configurations.

The inverse kinematics methods used were based on the CCD (Wang & Chen 1991) method and the NLP (Badler et al. 1987, Zhao & Badler 1994) method. The first method used a multiple chain version of the CCD method to simulate a skeletal structure. The second method involved the formulation of the IK problem as input to a nonlinear programming module, including the addition and implementation of various constraints. A naive dynamics-based approach was explored as a third method, which was based on a force-directed graph drawing algorithm. The parameters were chosen in an attempt to create extremely stiff springs with limited success in terms of it's ability to simulate the skeletal metaphor. Obviously, the focus of the previous work was on inverse kinematics. There remains scope for a study into dynamics-based techniques for skeletal animation applied to graph interaction.

In the previous work, the CCD & force-directed approaches resulted in a stretchiness of bones. This violates the prescribed skeletal metaphor and damages the perception that a physical entity is being manipulated.

The CCD method became very slow in some cases in the previous system. For a graph interaction system to be useful it must be useable in real-time, which was not always true of this approach.

The motion should also remain smooth and natural as is the intention of the skeletal metaphor. The NLP method provided jumpy motion and this limited its usefulness significantly.

Methods used in the previous work all have advantages and disadvantages both in terms of their satisfaction of the skeletal metaphor and in their usefulness in graph interaction. It would be desirable to use a technique that achieved and combined the advantages of the previous methods without the disadvantages. There are many techniques used for skeletal animation that could achieve this. There remains a large gap in the research for experimentation with new techniques for skeletal animation in this area, specifically dynamics-based techniques. However, it is also important to find methods that will be best suited to graph interaction. An ideal technique would both provide simulation of the skeletal metaphor and at the same time will be useful for graph interaction tasks. There is large potential for improvement upon the existing work by providing a highly responsive system that helps to increase the user's understanding of the graph by staying true to the skeletal metaphor.

3 Methods used for Dynamics-Based Skeletal Animation

With the graph modelled as a skeleton we need an algorithm to ensure that the skeletal constraints are maintained as the user modifies the positions of the nodes. The skeletal animation algorithms that we have used form a dynamics-based approach. These algorithms were adapted from the work of Jakobsen (2001) . Jakobsen's work was designed to control the physics systems in computer games where the major concerns are believability and speed of execution. One responsibility of the physics system is the simulation of articulated bodies. An articulated body is composed of solids and joints which define the range of motion allowed by the different parts of the body. The result is a structure that is very similar to that of a graph. Articulated bodies are often used to represent the skeleton structure of computer games characters (See Figure 1). As the physics system was designed for computer games, it provides simulation of a skeleton that appears natural to the viewer, and is fast, stable and well suited for interactive use. One way in which it has been used is for the dragging of articulated bodies around an environment. In many ways this is very similar to graph interaction. These algorithms are highly suitable for skeletal graph interaction because they have been specifically designed with interactive dragging of articulated bodies in mind.

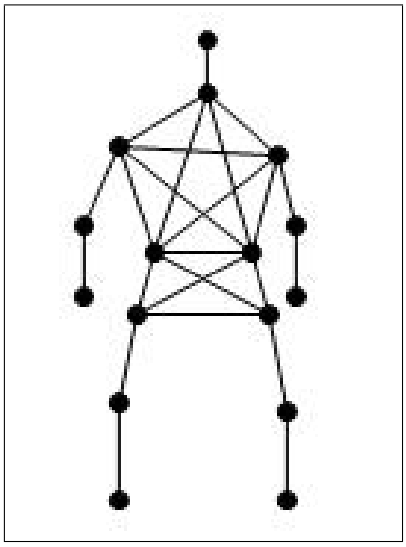

Figure 1: An Articulated Figure Representing a Character's Skeleton (Source: Jakobsen 2001).

Particle simulation is an important part of a physics system. The most commonly used method for particle simulation is Euler integration using Newton's law. In Euler integration each particle has two variables, its position x and its velocity v. In each time step the position and velocity values of the particle are updated. The new values x' and v' are given by:

$$x' = x + v \cdot \Delta t$$
$$v' = v + a \cdot \Delta t$$

where Δt is the time step, and a is the acceleration computed using Newtons law $f = ma$. The force f is the accumulated force acting on the particle by elements in the environment such as gravity.

The physics system (Jakobsen 2001) uses a Verlet integration scheme (Verlet 1967) for its particle simulation. Verlet integration schemes use a velocityless representation. The information stored consists of a particle's current position x and it's previous

position x^*. The new values using Verlet integration are given by:

$$x' = 2x - x^* + a \cdot \Delta t^2$$
$$x^* = x$$

This works similarly to Euler integration as $2x - x^* = x + (x - x^*)$ and $x - x^*$ provides an approximation of the velocity of the particle. Verlet integration is faster and more stable than Euler integration because it is harder for the velocity and position to come out of sync. This has allowed it to be useful for simulating molecular dynamics.

Common methods for handling collisions and contact of particles with the environment have a number of disadvantages that can make them unsuitable for interactive use. One method handles collisions by inserting springs at the penetration points. In this case, it is difficult to choose spring constants such that the system remains stable and accuracy is maintained. Another method goes back in time to the exact point of collision, where it handles the collision analytically before restarting the simulation. This method is likely to run very slowly when there are a lot of collisions making it impractical for interactive applications.

In the physics simulation collisions are handled by projection. This means moving the point perpendicularly out towards the collision surface as little as possible until it is free of the obstacle ensuring particles are kept inside the valid environment. This works well with the Verlet integration scheme as corresponding changes in velocity will be handled automatically and there is no need to directly cancel the velocity in the normal direction. It is also fast and simple to implement making it suitable for interactive use. Also, as we will see in the following sections, it works particularly well with constraint satisfaction and rigid or articulated bodies.

In a physics system particles are often connected by springs to model rigid bodies or cloth. The problems with this approach are once again associated with the choice of parameters. It is difficult to find a balance between weak springs that appear too elastic and strong springs that cause instability in the equations. It is also difficult to solve the differential equations used in this method.

The algorithms of the physics system model the connections between particles as infinitely stiff springs. These springs will instantly reach their rest length and are effectively more like sticks than springs. This makes them highly suitable for modelling bones of a skeleton. For each stick connecting two particles, the particles must be placed such that the distance between them is equal to the desired length of the stick. After the movement of a particle takes place due to a Verlet integration step the stick length may have become invalid. This constraint is validated by moving the particles away from each other or pulling them closer together, depending on whether the length of the stick joining them, is too large or too small. This is highly suited to an interactive application as it is a very fast and simple approach. The pseudo-code for satisfying the stick-length constraint is:

```
delta = x2-x1;
deltalength = sqrt(delta · delta);
diff = (deltalength-restlength)/deltalength;
x1 += delta*0.5*diff;
x2 -= delta*0.5*diff;
```

There are two constraints governing the system that must be satisfied. These are keeping particles within the environment and maintaining ideal stick lengths. The satisfaction of either of these constraints can lead to the invalidation of the other. To ensure that length and collision constraints are both satisfied, local iteration is performed, where the two constraint solvers are repeatedly run after each other. After a number of iterations the positions of the particles converge to a valid solution. This iterative method can be stopped at any time depending on the level of accuracy required, allowing for time/accuracy trade-off.

This physics system also has a further speed up mechanism if required making it even more suitable for interactive use. The costly square root operation that takes place in the stick constraint satisfier can be approximated. This approximation uses the fact that as long as the constraints are satisfied, or close to it, the result of the square root operation should be the rest length of the stick. Also, if the distance between the particles is already correct then no change is necessary. The square root function is approximated by its 1st order Taylor-expansion at a neighborhood of the squared rest length. The pseudo-code for satisfying the stick-length constraint becomes:

```
delta = x2-x1;
delta*=restlength*restlength/
  (delta · delta+restlength*restlength)-0.5;
x1 += delta;
x2 -= delta;
```

Rigid bodies are modelled as particles connected by sticks governed by constraints. Articulated bodies are modelled by connecting rigid bodies. It is possible to connect multiple rigid bodies by hinges and pin joints. Simply let two rigid bodies share a particle, and they will be connected by a pin joint. Share two particles, and they are connected by a hinge. This provides an intuitive mapping from the particle system to a skeleton where particles are mapped to joints and sticks are mapped to bones.

Angular constraints can add further realism to the particle system. One method given by Jakobsen uses additional constraints that are defined for each joint connecting two sticks. The constraints are that the distance between the unconnected particles of the sticks forming the joint cannot fall below some threshold. These constraints are enforced in the same way as the stick constraints in the case where the stick length is less than it's ideal length. This method effectively prevents some particles from coming too close together.

4 Implementation of Graph Interaction Environment using Dynamics-Based Skeletal Animation

We have implemented this physics system into a three dimensional graph interaction environment in Java3D. The graph is modelled as an articulated body with joints representing nodes and bones representing edges. Our implementation allows the user to drag the nodes of the graph around and the graph will respond according to the underlying physics engine.

As a node is dragged around the environment momentum is generated in the other nodes. The Verlet integration scheme is used to determine the

positioning of the nodes. There was a problem that no friction existed in the system and the graph would tend to float around endlessly, never settling. The equations governing the Verlet integration scheme were modified to introduce drag into the system. Jakobsen suggested the equation $x' = 1.99x - 0.99x^* + a \cdot \Delta t^2$ to introduce drag. However, this provided no noticeable difference. After some experimentation we used the equation $x' = 1.75x - 0.75x^* + a \cdot \Delta t^2$ which reduced the floating sensation and ensured the graph would settle quickly. This also gave the graph the impression of a rigid, physical structure as is the nature of a skeleton. We also don't have any need for gravity in our system, so the equation simplifies to $x' = 1.75x - 0.75x^*$.

A cube is used as the environment in which the graph must remain. Collisions and penetrations are projected onto the inside of the cube and when dragging a node the user cannot move this node outside the cube. Other nodes which move due to the momentum created by this drag and the stick constraints, will also stay within the environment. This allows us to restrict the movement of the graph within a small area and keeps the user's focus on this area. The use of this cube is optional and it when it is used only the outline is visible. This ensures that the user will not be not be distracted by it.

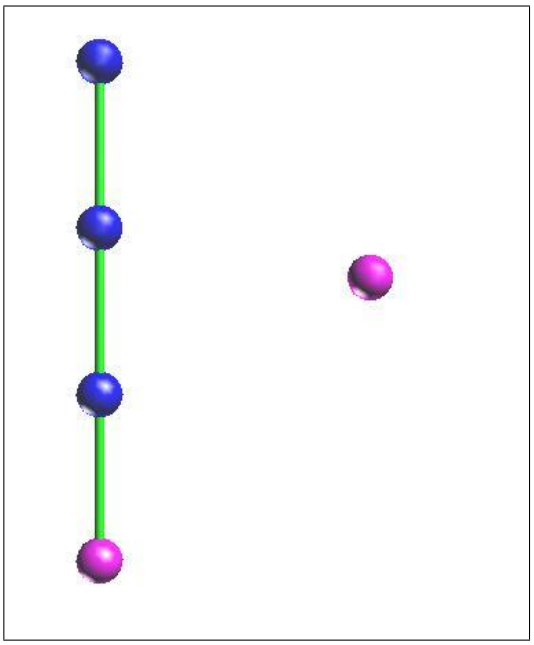

Figure 2: Initial Configuration Showing Desired Position of End-Effector.

As a node is dragged around, the edge length constraints are satisfied by the constraint solver (See Figures 2, 3). This ensures that as a node is dragged surrounding nodes will move closer or further away from the dragged nodes in order to maintain the constraint. This creates a natural feel in the system as the edges remain at constant length similarly to the bones of a skeleton. An example of the motion generated by the system can be seen in figure 8.

For each frame the iterative constraint solver is run to ensure collision and stick length constraints are satisfied. It also helps to preserve the user's mental map by providing a smooth animation from one configuration of the graph to the next. In practice we found that only a very small number such as 5

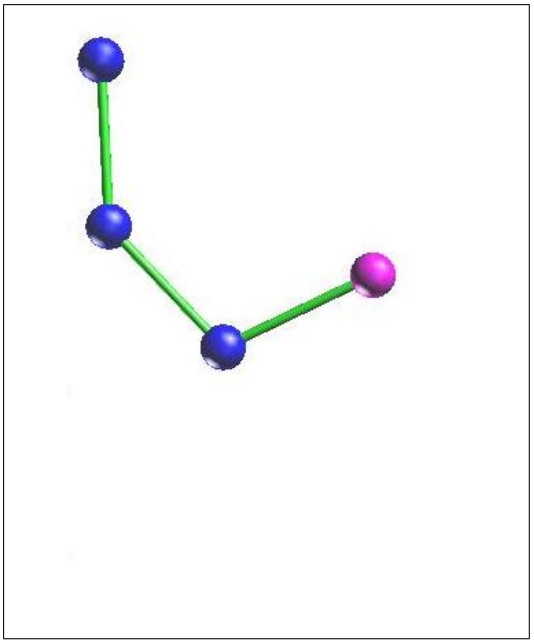

Figure 3: Movement of the Node to the Desired Position followed by the Constraint Solver Produces the Final Configuration.

iterations of the constraint solver were necessary. This is enough to reduce any large violations of the constraints. Any remaining violations in the constraints will be small and will not be noticed by the user. Also, the satisfaction of constraints will converge further over subsequent frames. Increasing the number of iterations does not provide significant improvement in terms of constraint satisfaction believability. With a large increase in the number of iterations performance can become an issue for large graphs, actually reducing the natural feel of the structure rather than increasing it.

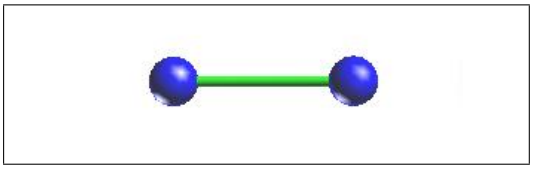

Figure 4: As neither nodes are fixed, the edge-length constraint is easily satisfied.

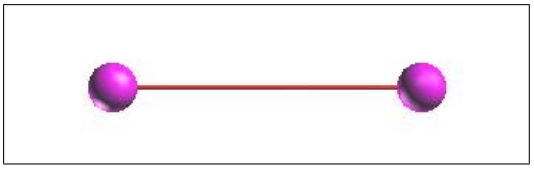

Figure 5: Both nodes are fixed. It is not possible to satisfy the edge-length constraint, as moving the nodes closer together would violate the fixed nodes constraint.

An additional aspect necessary for graph interaction but not handled in the physics system is that of fixed nodes in space. In any graph interaction environment it is necessary for the user to be able to fix nodes so that they will remain in the same

location. The user can select nodes to be fixed or can unfix nodes at any time. We have implemented this in our system by adding the constraint that the subset of nodes that is fixed cannot move. This is a hard constraint and is given priority over the other constraints so that the fixed nodes will not move from the set position at all. This introduces a problem in ensuring that the edge length constraints are satisfied, for example, if two nodes connected by a stick are fixed. If the user drags one of these nodes away the edge will increase in length and the constraint will be violated as the system cannot move either node in an attempt to satisfy the constraint (See Figures 4, 5).

To remedy this problem we have placed restrictions on how the user can drag nodes around. Movements of nodes that will violate an edge length constraint significantly are not allowed. This allows us to maintain some freedom of control for the user while still providing the natural feel of the skeletal structure. A threshold that measures the sum of violations across the entire graph is used to determine if the extent of a violation is acceptable. Once the maximum threshold is reached, movement will be limited to actions that reduce or maintain the current extent of violations. After reducing the extent of the violations, the user will then once again be able to make movements freely. This provides restrictions on the movement that more closely resemble a physical structure as is the nature of the skeleton. As a result, it is expected to provide a more natural feel and be more effective at maintaining the skeletal structure. In our system, we have provided a mechanism for navigating the graph with or without the threshold restrictions.

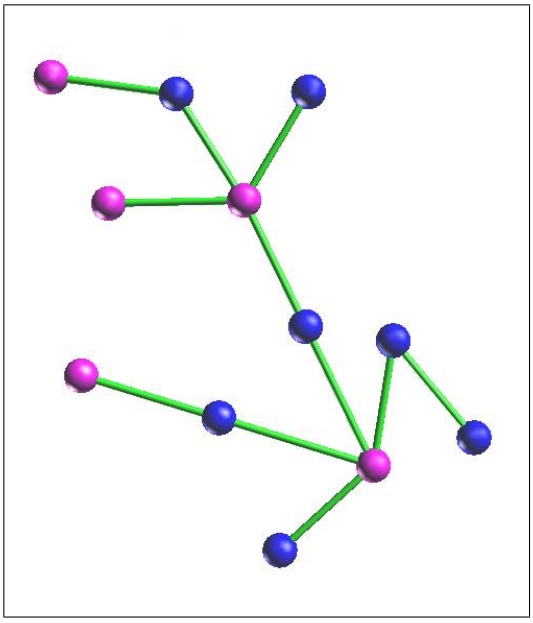

Figure 6: All constraints are satisfied. This is indicated by edges of standard width.

Our implementation can be used with or without the square root approximation. However, because of the fixed nodes in space the constraints of the system are less likely to be satisfied. When using the square root operation it is important to also use a low threshold to ensure that the constraints are at least close to being satisfied. The square root approximation does not seem necessary on smaller graphs but as the size of graphs increases it can be used to speed up the response to user actions.

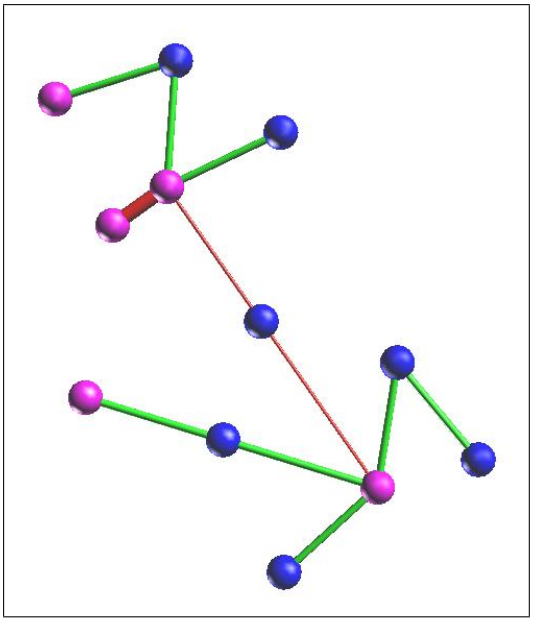

Figure 7: Some edge-length constraints are not satisfied. This is indicated by edges of varying width. Thin edges are too long while thick edges are too short. While it cannot be seen here, we also highlight violated edges in red to further emphasise which edge-length constraints are not satisfied.

We also provide visual cues as to where and by how much violations are occurring. A significant edge length violation will result in a change in colour of the edge from green to red. Also, the width of the edge will decrease if the edge is stretched and will increase if the edge is compressed (See Figures 6, 7). The visual cue of using blue for unfixed nodes and highlighting fixed nodes in pink is also provided to ensure the user is aware of the subset of fixed nodes.

When drawing a graph in our system we not only need to know the set of nodes and edges but also what the ideal length of each edge will be. In some domains, graph edges have associated weights and these can be used directly to determine the ideal edge lengths. In other cases determining the edge lengths may not be so straight-forward. A common aesthetic property desired in graph drawings is that of uniform edge-lengths, where all edges in the drawing have approximately equal lengths. In our system, the edge-lengths are provided as input along with the set of nodes and edges. The initial layout of nodes is produced randomly. The constraint solver then improves the drawing. The drawing can then be further improved by the user, as is the aim of our system. These interaction techniques could also be implemented together with an automated graph drawing system. The interaction mechanism could then be used in situations where the initial drawing provided is not easy to understand.

Our original definition of a skeleton does not specify any angular constraints. However, in terms of graph navigation, there is a possible benefit of implementing angular constraints as it would keep nodes from getting too close together. We have implemented angular constraints in our system by a similar method to that suggested by Jakobsen. We have applied forces between unconnected nodes. Unconnected nodes whose distance is less than a

threshold are repelled from each other. The forces on each node are summed in a way similar to force-directed methods and nodes are moved in the direction of the force. The calculation of forces and movement of nodes is performed within the constraint satisfaction loop. While this is not always an accurate way of providing angular constraints, it is fast and simple and provides a realistic feel most of the time. This method not only adds realism to the skeletal structure, it also provides aesthetic benefits. It prevents unconnected nodes from getting too close together and this results in good angular resolution. Edges emanating from a node will spread evenly around a node.

Part of the motivation for this paper was that the techniques used by Merrick were not highly successful in simulating the skeletal metaphor or increasing the user's understanding of the graph structure (Merrick 2002). Therefore we will demonstrate that we have improved upon the previous work by showing that the particle system is more effective in achieving these aims.

The CCD & force-directed approaches resulted in a stretchiness of bones. This violates the prescribed skeletal metaphor and damages the perception that a physical entity is being manipulated. The CCD method lead to a stretch in bones as a result of a cumulative error. In the particle system, any small errors will converge over subsequent frames while more iterations are applied. The force-directed technique resulted in very stretchy bones as a result of the naive approach in simulating the skeleton. The parameters were chosen in an attempt to create extremely stiff springs but it was not as effective as expected, as it is still possible for the springs to stretch. In the particle system, the edges are modelled as infinitely stiff springs, so their ideal length is reached instantly, minimising stretching. Generally, in the particle system, the only time when edge-length constraints remain unsatisfied, is when the set of fixed-nodes make it impossible to satisfy both fixed node constraints and edge-length constraints. The constraint solver ensures that the edge length constraint is always satisfied provided the set of fixed joints allows this to be possible. Edges that have violated the length constraint due to the nature of the fixed joints, will return to their original length once possible. The violation of bone-length constraints can be further reduced by using the threshold mechanism.

The CCD method became very slow in some cases in the previous system. For a graph interaction system to be useful it must be useable in real-time, which was not always true of this approach. The particle system is designed to be very fast in order to be useful for interactive purposes. Verlet integration is a fast method of simulating particle motion. Satisfying environmental constraints by projection is also very fast as is satisfying edge-length constraints. It also allows for a speed versus accuracy trade off by adjusting the number of iterations per frame, without accumulating visually obvious errors. The square-root approximation can also be used to further speed up the system's response to user actions.

The motion should also remain smooth and natural as is the intention of the skeletal metaphor. The NLP method provided jumpy motion and this limited its usefulness significantly. Conversely, the motion provided by the physics system is smooth and natural, helping the user to quickly and easily understand the graph structure.

None of the previous methods implemented any form of angular constraints. Angular constraints not only add a greater level of realism to the skeletal structure, they also greatly improve the angular resolution of the resultant graph drawing. By providing a technique with angular constraints we provide our system with an additional advantage over the previous techniques.

The physics system provides the advantages of all three previous methods while also providing some additional advantages. It also has none of the disadvantages of the previous techniques. It provides satisfaction of the constraints of a skeleton while ensuring the naturalness of the skeletal motion and fast speed of execution. The particle system provides a fast, accurate simulation of the skeletal metaphor that increases the users ability to understand the graph structure.

5 Conclusion / Future Work

We have introduced the particle system as a method for skeletal animation that can be used for graph interaction tasks. As the skeleton is a natural structure, and this method was designed for interactive use with the dragging of joints in mind, it is quite suitable for providing a graph interaction environment that feels natural and intuitive. We made the method even more more suitable for this purpose by adding features specifically designed to aid graph interaction such as fixed node constraints, an error threshold and visual cues. The adapted particle system improves upon the previous techniques used for skeletal animation in graph interaction. This shows that our new technique has had a positive effect on the simulation of the skeletal metaphor in graph interaction.

The satisfaction of the constraints of a skeleton while ensuring the naturalness of the skeletal motion and fast speed of execution, provides a fast, accurate simulation of the skeletal metaphor that increases the user's ability to understand the graph structure.

This work adds to the field of graph interaction by providing a natural system of navigating a graph with the ability to increase the user's understandability of the graph structure.

There is still a significant need for empirical results indicating the advantages and disadvantages, if any, of using a skeletal metaphor over other types of interaction with graph visualisations. There remains a gap in the research that could be filled with a more comprehensive user study in order to produce solid evidence of the usefulness of the skeletal metaphor. It is also important to evaluate the usefulness of each technique in simulating the skeletal metaphor.

The only method of interaction with the system presented in this paper, and the system described by Merrick (Merrick 2002) was through the use of a standard two dimensional mouse. Other devices for interaction with three dimensional displays could provide a more natural interface between the user and the system, complementing the skeletal metaphor. In particular, haptic interaction devices not only provide a method for interaction in three dimensions but use force feedback techniques to give the user an actual physical feel of what they are interacting with. The benefits of haptic devices have been explored (Hinckley 1996) and these devices have been shown

to be quite useful for a range of applications. It remains to be determined, when using a graph interaction system utilising the skeletal metaphor, what additional benefits would be provided by using three dimensional input devices such as haptic devices. One of many possible benefits would be to allow the user to more easily determine when a threshold has been reached through the use of force-feedback.

The work of Ware and Franck has shown (Ware & Franck 1994) that many of the benefits of three dimensional visualisations are further realised when 3D display devices such as a stereoscopic, hand-tracked 3D display are used. The use of the skeleton metaphor with such devices is therefore likely to see even more benefits.

This paper has not only contributed to the field of graph interaction but have also opened up many possibilities for future work in this area. Such research is likely to provide an important contribution in the fields of both graph visualisation and human computer interaction in the future.

References

Di Battista, G., Eades, P., Tamassia, R. & Tollis, I. (1994), Algorithms for Drawing Graphs: an Annotated Bibliography, *Computational Geometry: Theory and Applications, 4 (5), pp. 235-282.*

Di Battista, G., Eades, P., Tamassia, R. & Tollis, I. (1999), Graph Drawing: Algorithms for the Visualization of Graphs, Prentice-Hall, Englewood Cliffs, NJ.

Esposito, C. (1988),Graph Graphics: Theory and Practice, *Computer Mathematics Applications, 15(4), 247-253.*

Dengler, E., Friedell, M., and Marks, J. (1993),Constraint-Driven Diagram Layout, In Proceedings of the 1993 IEEE Symposium on Visual Languages, pages 330–335, Bergen, Norway.

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

Batini, C., Furlani, L., and Nardelli, E. (1985),What is a Good Diagram? A Pragmatic Approach, In Proc. 4th Internat. Conf. on the Entity-Relationship Approach, pages 312-319.

Ding, C., and Mateti, P. (1990),A Framework for the Automated Drawing of Data Structure Diagrams, *IEEE Transactions on Software Engineering, SE-16, (5), 543–557.*

Purchase, H. C., Cohen, R. F., and James, M. I. (1997),An experimental study of the basis for graph drawing algorithms, Journal of Experimental Algorithmics (JEA), 2:4.

Purchase, H. C., McGill, M., Colpoys, L., Carrington D. (2001),Graph drawing aesthetics and the comprehension of UML class diagrams: an empirical study, *In Australian symposium on Information visualization, pages 129 137. Australian Computer Society, Inc.*

P. Gould and R. White (1986),Mental Maps, Allen and Unwin Inc., Winchester, Mass. 01890, USA

Eades, P., Lai, W., Misue, K., and Sugiyama, K. (1991),Preserving the mental map of a diagram, *in Proc. of Compugraphics, pp. 24-33.*

Misue, K., Eades, P., Lai, W., and Sugiyama, K. (1995),Layout adjustment and the mental map, Journal of Visual Languages and Computing, 6(2):183–210.

Friedrich, C. (2002),Animation in Relational Information Visualization, *PhD Thesis, The University of Sydney.*

Bruβ, I., and Frick, A. (1996),Fast Interactive 3D graph visualisation, Lecture Notes in Computer Science, 1027:99—110.

Merrick, D. (2002), Skeletal Animation for the Exploration of Graphs, *Honours Thesis, School of Information Technologies, The University of Sydney.*

Jakobsen, T. (2001), Advanced Character Physics, *TechReport, IO Interactive*

Verlet, L. (1967), Computer experiments on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules, *Phys. Rev., 159, 98-103.*

Herman, I. and Melanon, G. and Marshall, M. S. (2000), Graph visualization and navigation in information visualisation: a survey, *IEEE Transactions on Visualization and Computer Graphics, 6(1):24–43.*

Ware, C. and Hui, D. and Franck, G. (1993), Visualizing object oriented software in three dimensions, *In Proc. IBM Centre for Advanced Studies Conf., CASCON.*

Ware, C. and Franck, G. (1994), Viewing a graph in a virtual reality display is three times as good as a 2-d Diagram, *In IEEE Conference on Visual Languages, pp. 182-183.*

C. Ware and Franck, G. (1996), Evaluating stereo and motion cues for visualizing information nets in three dimensions, *ACM Transactions on Graphics (TOG), 15(2):121-140.*

Cockburn, A. and McKenzie, B. J. (2001), 3d or not 3d? Evaluating the effect of the third dimension in a document management system, *In CHI, pages 434-441.*

Nielsen, J. (1998), 2d is better than 3d, *http://www.zdnet.com/devhead/alertbox/981115.html*

Wang, L. C. T. and Chen, C. C. (1991), A combined optimization method for solving the inverse kinematics problem of mechanical manipulators, *The International Journal of Robotics Research.*

Badler, N. and Manoochehri K. and Walters, G. (1987), Articulated figure positioning by multiple constraints, *IEEE Comput. Graph. Appl. 7(6), pages 28-38.*

Zhao, J. and Badler, N. I. (1994), Inverse kinematics positioning using nonlinear programming for highly articulated figures, *ACM Transactions of Graphics (TOG), 13(4):313-336.*

Hinckley, K. (1996), Haptic Issues for Virtual Manipulation, *Ph.D. dissertation, University of Virginia, Dept. of Computer Science.*

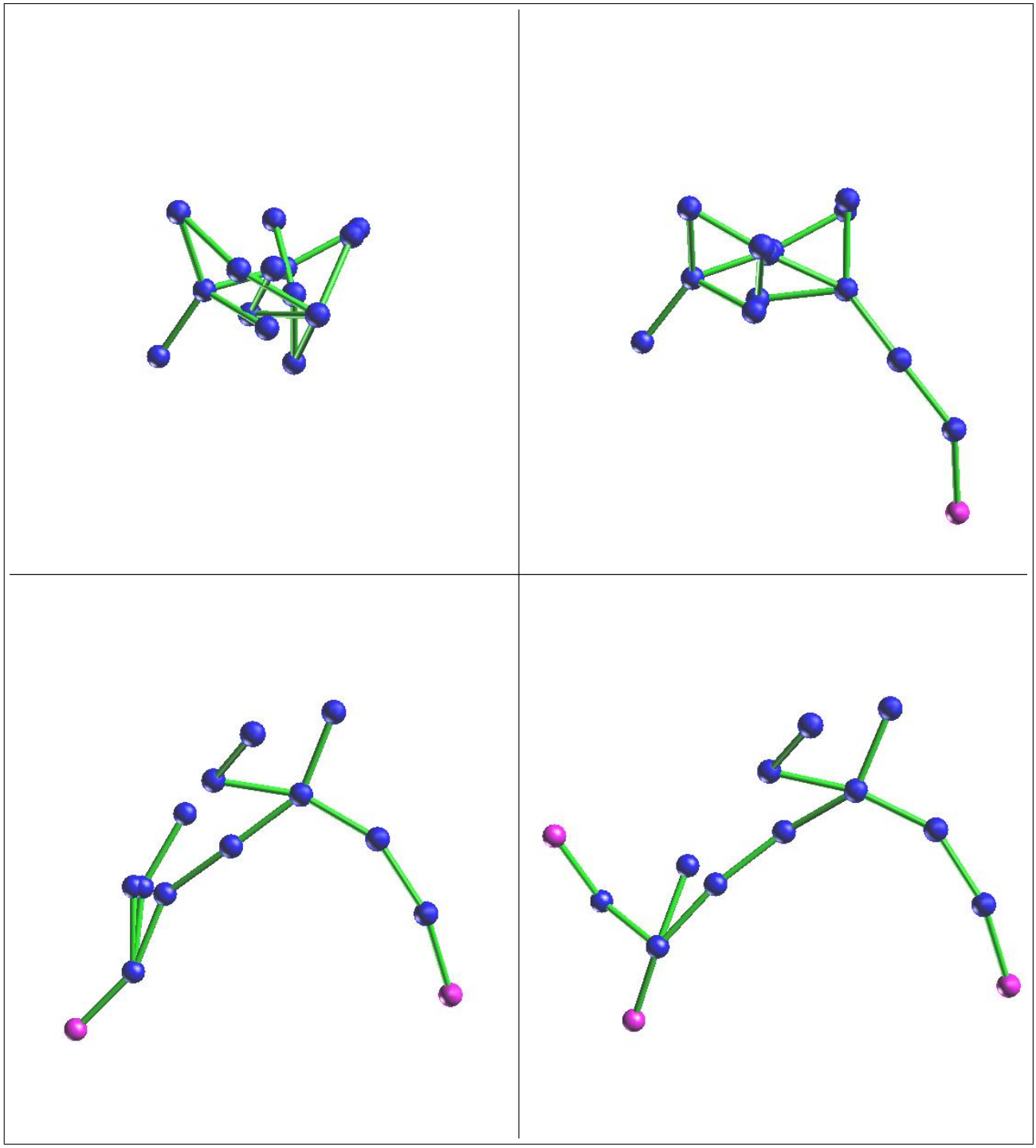

Figure 8: Dragging nodes using our dynamics-based skeletal graph interaction system. By ensuring the constraints are satisfied it quickly produces a much better graph drawing and allows us to improve our understanding of the graph structure.