Skeletal Animation for the Exploration of Graphs

Damian Merrick and Tim Dwyer

School of Information Technologies
University of Sydney
Sydney, NSW, 2006, Australia

{dmerrick,dwyer}@it.usyd.edu.au

Abstract

The topic of skeletal animation and its associated techniques have previously been applied in the area of animating computer-generated characters for motion pictures and computer games. This paper investigates the use of similar techniques in the scope of exploring three-dimensional visualisations of graphs.

A system is discussed which, after generating an initial 3D layout for a graph, creates a structural "skeleton" of the graph and allows a user to push, pull and drag nodes of the skeleton in order to manipulate the layout. Skeletal animation is used to smoothly animate the graph layout according to the movement applied by the user as well as various underlying constraints forced on the graph's skeleton. Several algorithms for performing this skeletal animation are proposed, and evaluated to determine the relative benefits and disadvantages of each.

Keywords: information visualisation, graph interaction, skeletal animation

1 Introduction

Research into three-dimensional graph visualisation has raised various issues about user interaction with drawings of graphs. Past work has shown that the amount of information one can perceive can be significantly larger when a 3D visualisation is used in place of a 2D visualisation, given sufficient spatial cues [Ware, Frank, 1994, Ware, Hui, Frank, 1993]. Despite this, however, there seems to be little doubt that the extra dimension also adds a factor of difficulty in navigating the visualisation space. This navigation difficulty can increase the time needed by someone analysing a drawing of a graph to understand the information it portrays. In turn, this reduces the benefit that may otherwise be realised by utilising three dimensions in a visualisation.

If an effective paradigm for interaction and navigation is used, however, some of this difficulty may be overcome. Cockburn and McKenzie performed a user study comparing the effectiveness of a 3D visualisation versus a 2D visualisation in a document management system [Cockburn, Mackenzie, 2001]. The results of this study indicated no significant benefit in using the 3D visualisation, but the authors considered their 3D system relatively naïve, seemingly not possessing sufficient perceptual cues to provide a believable 3D environment on a 2D display device. Provision of more cues to enhance the realism and intuitiveness of a 3D visualisation may help to improve this situation.

This paper explores the possibility of a physical metaphor for navigation with a graph visualisation. A physical metaphor allows users to relate the interaction mechanism with phenomena that occur in the real world, providing perceptual cues that will increase the intuitiveness of the visualisation system. In using a physical metaphor, we may facilitate a user's manipulation of a visualisation in order to identify patterns and investigate inherent structural characteristics of the underlying information. This will tentatively provide us with a useful method of interaction which may be put to use as part of a graph visualisation tool. With such a tool, users may create a good layout of a graph even when the initial automatic layout provided is hard to understand.

2 Skeletal Animation

Skeletal animation is a concept that has been implemented to provide the illusion of natural motion for computer-generated characters in films and computer games. From a simplistic point of view, a skeleton may be thought of as a collection of rigid bones connected together by revolute joints (points of articulation that allow the rotation of connected bones about one or more axes). This structure is one that we, as humans, observe and manipulate constantly in the real world. Thus it would seem reasonable to suggest that skeletal motion, simulated adequately, would be a natural and realistic physical metaphor. One may also observe a convenient mapping between a skeleton and a graph. Given a graph G(V,E), let the set of skeletal joints J map to the set of vertices V, and the set of bones B map to the set of edges E. We are particularly interested in the case where G is a higher level abstraction of another graph G'. Using such a mapping one may produce a skeleton which is analogous to a defined subset of a graph of interest.

In order to make use of this skeleton to interact with its associated graph, several issues must be addressed:

1. An initial three-dimensional graph layout must be generated.

2. A subgraph must be selected for representation as the skeleton. Ideally, this will be in some way representative of the structure of the graph (for example, a cluster tree, or the critical path in a PERT diagram).

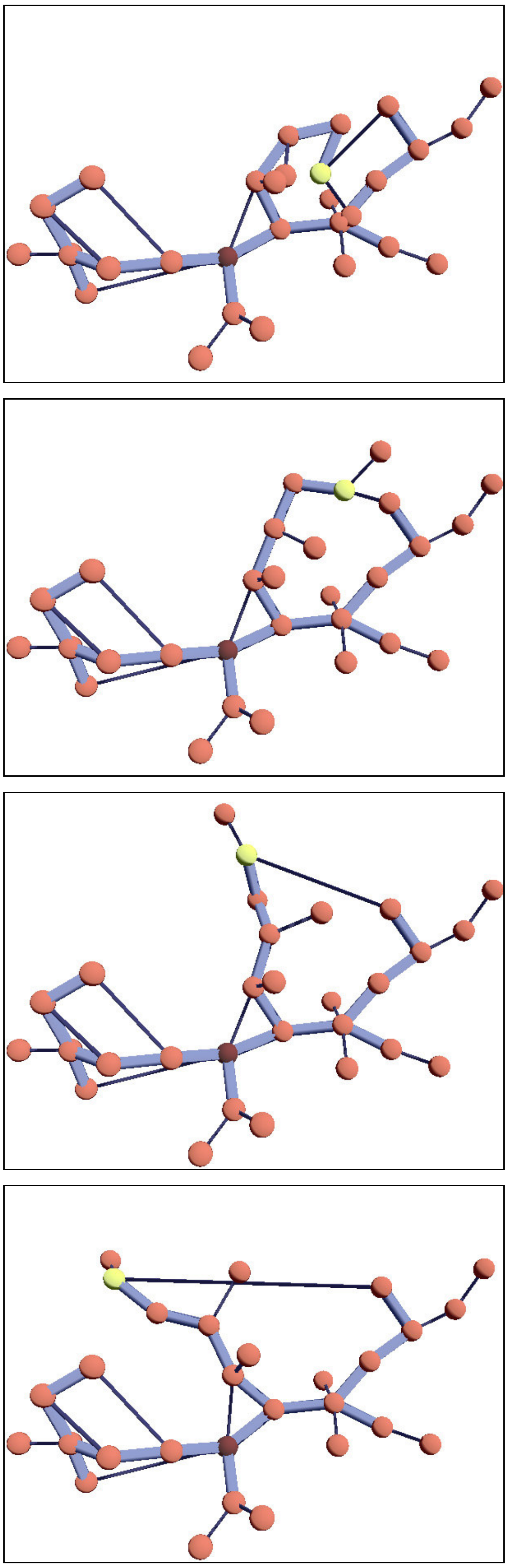
Figure 1: Successive screenshots from the sGraph system

3. Motion must be generated for the graph based on user interaction with the skeleton.

We have developed a system called sGraph that attempts to address these issues. The system displays a 3D drawing of an undirected graph (a multi-layer layout approach is used [Walshaw, 2000]), and chooses a skeletal subgraph as a spanning tree of the graph with a percentage of nodes removed according to an importance metric (similar to the Strahler metric described by Herman et al. [Herman et al., 1999]). This skeletal subgraph is represented visually by thicker edges in the 3D drawing. The user can push, pull and drag nodes of the skeleton by using the mouse, and the rest of the graph reacts to the input through application of a skeletal animation algorithm. Example screenshots showing a node being dragged around by a user are presented in Figure 1.

The main goal of this paper is to explore such skeletal animation algorithms, and their application to graph interaction. Substantial research has been performed into the motion of articulated structures, originally in the field of robotics, and more recently in computer animation. This research can generally be divided into two categories; kinematic and dynamic. Kinematics is the study of the geometry of motion, where dynamics typically looks at simulating systems of forces to produce an outcome. Much of the literature concerns kinematic algorithms, and so this paper focuses primarily on these.

A classic problem in this area involves controlling the operation of a number of mechanical joints on a robot arm, in order to move the "hand" of the arm to some desired location. Figure 2 and Figure 3 show an example of this, with a chain of four joints connected by three bones, in initial and desired configurations, respectively. An analogue of this can be created in a

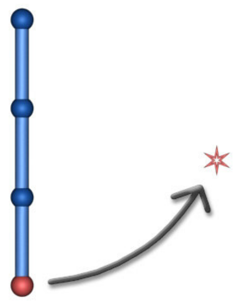
Figure 2: Initial configuration of skeletal chain

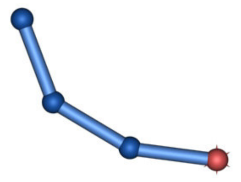
Figure 3: Final configuration of skeletal chain

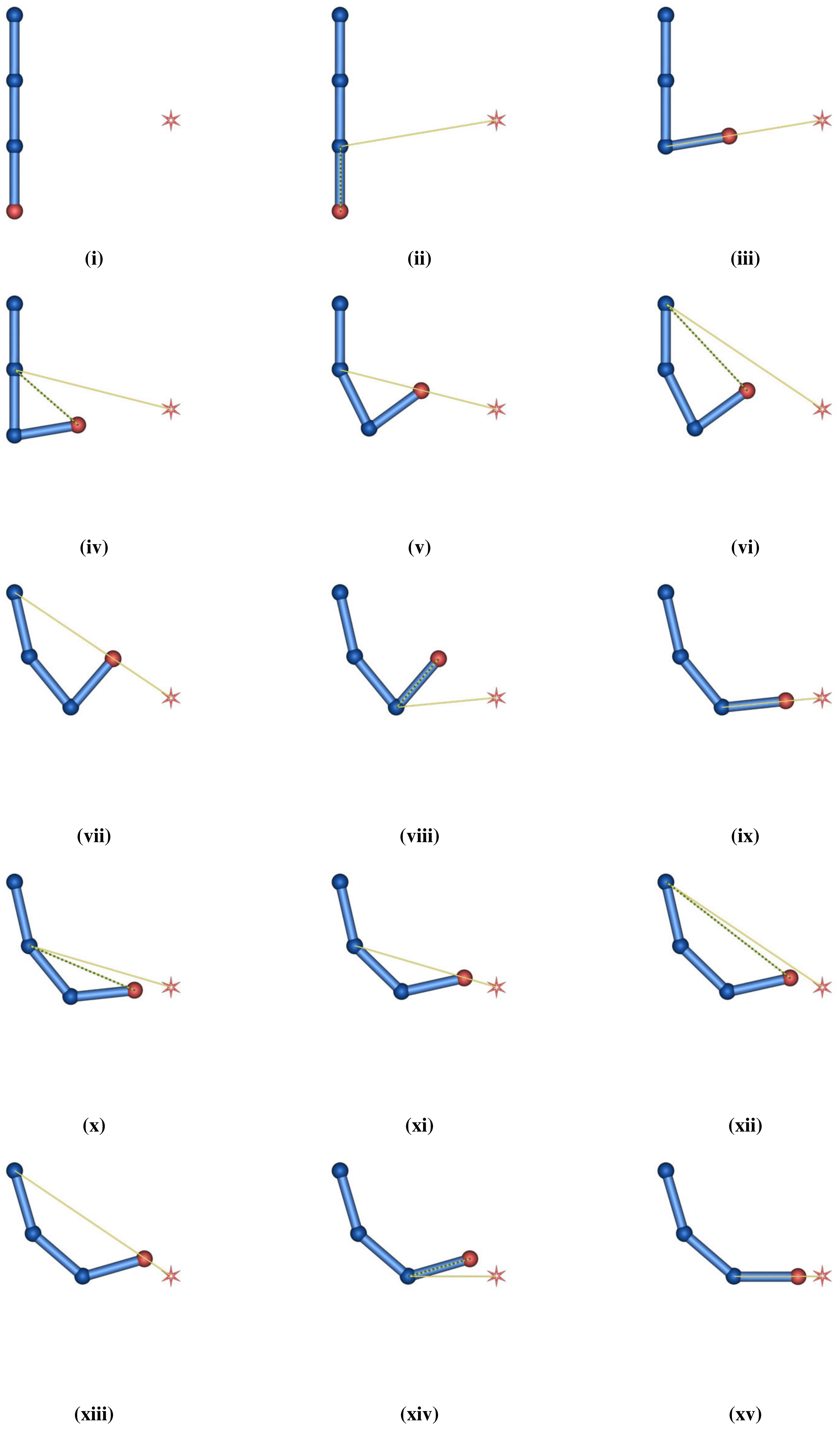

Figure 4: An example of the CCD algorithm executing over several iterations

graph visualisation environment. That is, a user dragging a node of a graph's skeleton from one position in space to another.

The difficulty in solving this problem lies in the fact that in many cases there are an infinite number of configurations of the arm which produce the desired solution. Several methods have been investigated to deal with this, including the pseudoinverse method [Klein, Huang, 1983], the Jacobian transpose method [Sciavicco, Siciliano, 1987, Welman, 1993, Wolovich, Elliott, 1984] and Cyclic Coordinate Descent (CCD) [Wang, Chen, 1991, Welman, 1993]. All of these methods are able to solve the aforementioned problem, although the pseudoinverse and Jacobian transpose methods suffer in some circumstances from problems with singular matrix configurations. The CCD method is a heuristic-based approach which does not have these problems, and has been adopted for use in this paper.

The approach taken by CCD is to solve a localised optimisation problem at each bone in the articulated chain. At each step of the algorithm the bone in question is rotated (and with it all bones and joints below it in the chain) such that the distance between the end-effector (the last joint on the chain) and the desired position of the end-effector is minimised. This process starts from the bottom bone in the chain and works towards the top, then repeats over again, until the end-effector is within some threshold distance of its desired position. Figure 4 gives a visual example of the CCD algorithm executing over a number of iterations. In this figure, the lightly coloured solid line indicates the line from the current joint to the target position and the darkly coloured dotted line indicates the line from the current joint to the end effector.

Other research has seen the entire problem presented as a nonlinear optimisation [Badler et al, 1987, Zhao, Badler, 1994]. An objective function is formulated, based on the distance of the end-effector from its target, constraints are applied, to ensure bones remain rigid, and an optimisation algorithm executed to obtain a valid configuration. We have investigated such an algorithm for use in a skeletal graph interaction mechanism, though the initial results produced were not highly intuitive [Merrick, 2002].

3 Algorithms for Skeletal Animation in Graphs

Three algorithms were formulated and implemented for performing skeletal animation in graph visualisation systems – One based on the CCD method, another based on the nonlinear optimisation method, and the third a naïve dynamics-based implementation implemented as a natural extension to an existing 3D graph visualisation system.

These methods were compared in an informal user study, to evaluate their effectiveness relative to each other. We define effectiveness as the ability of a given technique to successfully convey the skeletal metaphor. This was measured primarily by subjective questions given to study participants and backed up by quantitative measurements of distances moved by nodes during the study session. The questions posed to the participants were as follows:

A. How well does the technique preserve the structure of the skeleton? (Structure is preserved when the lengths of bones remain constant)

B. How intuitive is the system (The way bones in the skeleton react to dragging around a node)?

C. While moving the structure was it easy to follow the skeleton?

Question A aimed to measure the satisfaction of the bone length constraint. Question B was posed to measure the relative subjective intuitiveness of each technique. Question C aimed to determine how well a user's mental map was preserved. See the literature for further discussion on mental maps [Eades et al., 1991, Misue et al., 1995]).

A brief overview of each of the methods and their advantages and disadvantages is given here. Full details of the algorithms and the evaluation performed on them are given in [Merrick, 2002].

3.1 Multiple-Chain Cyclic Coordinate Descent

As stated earlier, the CCD method provides a reasonable mechanism for controlling the configuration of a single articulated chain, given an end-effector joint on the chain and a desired position for that end-effector. It is important to note, however, that in our situation, we are dealing with an articulated structure that is potentially a tree or even a general graph. In the previous problem, the top-most joint of the chain was considered fixed. We would also like a user to be able to specify any number of points in the skeletal structure that will remain in a fixed position. By allowing the user to fix nodes, part of the structure can be forced to remain stationary in space while another part is manipulated. Without such a facility, the whole structure would drift in space and the benefit of the skeletal metaphor would be minimal. In both the sGraph and Wilma systems, users may fix and unfix nodes at any time (nodes are not fixed permanently). The multiple-chain CCD method allows this by applying CCD successively over multiple articulated chains extracted from the skeletal structure.

Consider the situation shown in Figure 5. The end-effector is the bottom-most joint in the image, and the other three outer-most nodes in the graph are fixed. We desire to move the end-effector to the position denoted by the star.

The problem has suddenly increased in complexity from the single chain we were dealing with in proposing the CCD algorithm. We are presented with

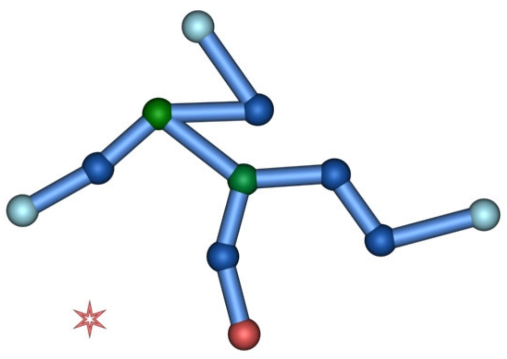

Figure 5: An acyclic graph structure of bones and joints

several questions that must be answered if we are to follow the steps prescribed in the CCD algorithm:

- Given that we want to traverse the structure in some way from base to root, and that the end-effector defines the base, what is to be considered the root of the structure?
- During the traversal, which path do we take when coming to a joint that has two or more parent joints?
- If we have adopted a certain fixed point as the root of the structure, and we are performing a rotation of a particular joint, how are the other joints in the structure affected? Intuitively all joints further from the root than the rotating joint should be rotated. This presents issues, however, in that some of these joints may be fixed, and thus should not be moved in any way.

Notice that in a skeletal structure such as this, articulated chains may be constructed from the end-effector to each individual fixed joint, ignoring joints not present on the direct path between them. Doing this with the Figure 5 example would give three chains which could then be solved individually using the CCD algorithm. Assuming the sum of the lengths of the bones in each chain is sufficient, the algorithm would provide configurations for each of the three chains that result in the end-effector being moved to the desired position.

This will clearly not be sufficient, however – There exist joints in the skeleton which are shared by more than one of these individual articulated chains. Although the successful execution of the CCD algorithm on each chain will ensure the end-effector is placed in the desired position (or at least within a certain acceptable threshold distance), the positions of the other shared joints are not guaranteed to be the same between any two of the given output configurations.

We may, however, choose one of these chains as the controlling chain (Figure 6). Executing the CCD

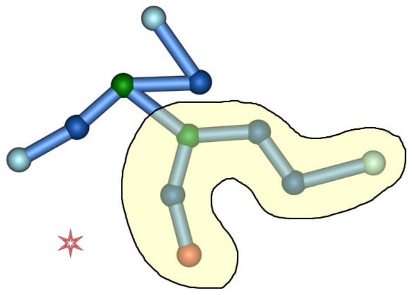

Figure 6: A controlling chain is chosen

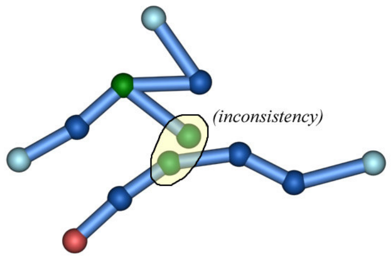

Figure 7: A solution is generated for the controlling chain

algorithm on this controlling chain gives a new set of coordinates for each joint in the chain (Figure 7). Notice that there are currently two visual representations of the single joint adjoining the controlling chain to the rest of the graph. This is obviously not desired, but one may realise that this new problem may be formulated again as a goal-directed motion problem on the rest of the graph. That is, we select the common joint as a new end-effector that we desire to move into the position specified by the CCD solution of the controlling chain. Since we already have a solution for the original controlling chain, we may ignore it and deal solely with the remaining part of the graph.

In this remainder of the graph, two fixed joints are present, and as with the original situation we construct a controlling chain from our new end-effector to one of the two fixed points, as illustrated in Figure 8. Once more, we generate a solution for this chain ignoring the rest of the graph, and the two representations of the common point between the new and old chains now lie in the same position in space, restoring consistency to that part of the graph (Figure 9).

One more step is necessary, however, as the second CCD solution leaves two distinct representations of the joint shared between the newest controlling chain and the unsolved portion of the graph. In the unsolved subgraph we are left with only one fixed point, so we build a chain from the inconsistent joint (as our new end-effector) to this remaining fixed joint (Figure 10). A final CCD solution on this chain aligns the two representations of the new end-effector, and we have a complete, consistent solution for the skeleton that has

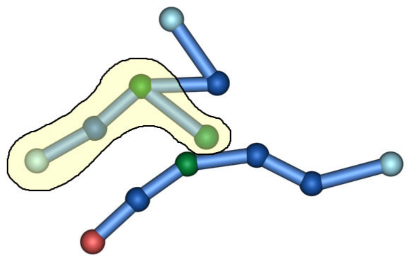

Figure 8: A controlling chain is chosen from the remainder of the graph

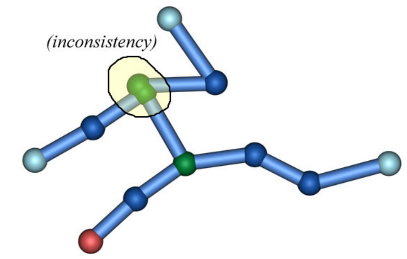

Figure 9: A solution is generated for the new controlling chain

brought the original end-effector into the position desired. Figure 11 illustrates the final solution for our example.

Of the three algorithms evaluated, results from the user surveys indicate the multiple-chain CCD method to be the most useful in presenting a skeletal interaction metaphor. In general, users believed it to be the most intuitive (Question B) and the most successful in maintaining their mental maps over successive configurations of the skeleton (Question C). One significant flaw is that it is possible for error in positioning of end-effectors to accumulate, allowing bones to stretch a substantial distance from their original length (Question A).

3.2 Nonlinear Programming

The nonlinear programming method takes the aforementioned approach of formulating the skeletal system as a set of model parameters, constraints and an objective function. These are passed on to a nonlinear optimisation module, which attempts to produce a new set of model parameters that minimises the objective function.

To address the problem in a form suitable for nonlinear optimisation, there are three things we must specify:

- A set of model parameters (the set of values for which an optimal configuration is desired)
- An optional set of constraints on the model parameters that must be satisfied in optimising the system

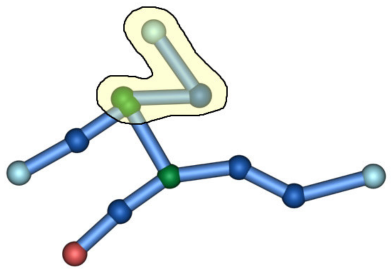

Figure 10: The last fixed joint is chosen to construct a controlling chain

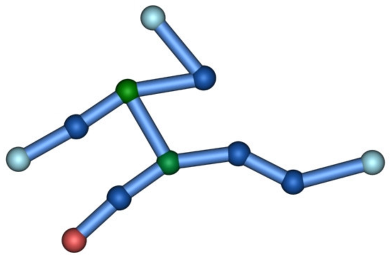

Figure 11: A solution is generated, yielding a complete solution for the skeleton

- An objective function to be minimised or maximised

The set of model parameters, then, is comprised of the joint variables that we wish to be modified to produce a new chain configuration. In the three-dimensional case that we are interested in we may represent each joint as a set of spherical coordinates and thus construct a model parameter set J:

$$J = \{r_1, \theta_1, \phi_1, r_2, \theta_2, \phi_2, \ldots r_n, \theta_n, \phi_n\}$$

In such a representation, the set of coordinates (r_i, θ_i, ϕ_i) are the spherical coordinates of the *i*th joint in the chain j_i.

A representation in spherical coordinates means that if we want to evaluate the position in Cartesian coordinates of the end-effector (the *n*th joint), we must traverse the chain entirely. On the other hand, it gives us a convenient representation where each set of values effectively specifies the length (r) and orientation (θ, ϕ) of each bone in the articulated chain.

In optimising the set of model parameters, small increments are typically applied to parameters over a number of iterations. In this way the representation that we have chosen is useful – Each of the increments applied corresponds to a rotation of a joint or a change in the length of a bone, rather than an arbitrary translation in the Cartesian coordinates of a joint.

Having noted this property, recall that changes in bone length are undesirable. In the given representation though, without further specification, they will be freely modifiable by the optimisation algorithm. There are two possible ways in which we may overcome this problem:

1. Remove the r_i values from the set of model parameters, so that only the angles between joints may be changed.
2. Add constraints to the system which encourage the differences between the original and actual lengths of each bone in the system to approach zero.

The first method may be the desirable option for the single articulated chain case as it absolutely ensures that there will be no changes in bone length. In order to solve more complex systems, however, it may be prudent to allow some slight changes in bone length to allow the optimisation algorithm to more easily find a solution. In this case the second approach is very appealing, as it provides a mechanism for bringing the change in bone length as low as possible without enforcing a strict limit. Our system used the first approach, for ease of implementation.

Now that we have defined our model parameters and a potential set of constraints, it remains to specify our objective function. As with the single-joint optimisation problem in CCD, we are looking to minimise the position error of the end-effector node. That is, we have an objective function E such that:

$$E = (x_n - x_D)^2 + (y_n - y_D)^2 + (z_n - z_D)^2$$

Where (x_n, y_n, z_n) is the actual position of the end-effector joint (calculated down the chain from the spherical coordinates using the previously detailed method) and (x_D, y_D, z_D) is the goal or desired position of the end-effector. Our objective function is then calculated as the square of a standard Euclidean distance metric between the current and goal positions of the end-effector. We wish to minimise this function, in order to bring the end-effector joint as close as possible to the desired position.

With the definition of the objective function, model parameters and constraints we have our required inputs into the nonlinear optimiser. We may think of the optimiser as a "black box" to which we provide these inputs, and from which we obtain the final output. A typical nonlinear optimisation module will be an iterative process requiring the provision of a method to evaluate the objective function at each iteration, and will potentially require a method to calculate the partial derivatives of that objective function in addition (the change in the objective function according to a change in each individual model parameter).

Our implementation used the BFGS nonlinear optimisation method [Broyden, 1970, Fletcher, 1970, Goldfarb, 1970, Shanno, 1970]. This algorithm attempts to find a local minimum of an objective function by taking steps in an error space which is based on partial derivatives of the objective function. Full details of the algorithm can be found in the literature.

Results from this method were rather discouraging, as the motion produced was jumpy (Question C) and non-intuitive (Question B), compared with the smooth transitions of the other two algorithms. However, the nonlinear programming method did have the advantage of being able to keep bone lengths constant (Question A). This is due to the fact that these lengths need not be specified in the model parameters (and hence will not be changed).

3.3 Physical Force Simulation

A third algorithm explored a simple dynamics-based approach to providing a skeletal metaphor. This approach was implemented as an addition to the 3D graph visualisation package Wilma [Dwyer, 2000, Dwyer, 2001]. Wilma makes use of a force directed algorithm for the layout of graphs, representing edges as springs connecting nodes. For comparison, Figure 12 and Figure 13 show the same graph laid out in sGraph and Wilma systems, respectively.

Wilma's layout algorithm is a spring embedder, representing edges as springs which bring adjoined nodes together, and applying repulsion forces between each pair of nodes in the graph. The attraction and repulsion forces $\overrightarrow{F_{spring}(n_1,n_2)}$ and $\overrightarrow{F_{repulsion}(n_1,n_2)}$ between two nodes n_1 and n_2 are defined as follows:

$$\overrightarrow{F_{spring}(n_1,n_2)} = \overrightarrow{V(n_1,n_2)}\frac{\left(\left|\overrightarrow{V(n_1,n_2)}\right| - l_{init}\right)C_{spring}}{\left|\overrightarrow{V(n_1,n_2)}\right|}$$

$$\overrightarrow{F_{repulsion}(n_1,n_2)} = \overrightarrow{V(n_1,n_2)}\frac{C_{repulsion}}{\left|\overrightarrow{V(n_1,n_2)}\right|^2}$$

where $\overrightarrow{V(n_1,n_2)}$ is a vector from n_1 to n_2, l_{init} is the relaxed (natural) edge length, C_{spring} is the spring "stiffness" constant and $C_{repulsion}$ is the repulsive force strength constant.

To achieve a skeletal metaphor within this system, the spring stiffness C_{spring} was increased for edges belonging to the skeleton. This was done with the aim of prohibiting those skeletal edges from stretching, creating the illusion of a skeleton in an otherwise elastic simulation.

The actual implementation of this method still allowed a very large amount of stretch in bones, however (Question A). Despite this, it seemed to appeal to users as a reasonable technique for graph interaction, and was reasonably intuitive (Question B). Smooth transitions were produced, due to the force-based layout, which aided retention of the mental map (Question C).

3.4 Relative Effectiveness of Techniques

Results from the user study were collected and aggregated to form relative rankings of the techniques according to the three given questions. The results of this process are given in Table 1.

	Question A	Question B	Question C
CCD	2nd	1st	1st
NP	1st	3rd	3rd
PFS	3rd	2nd	2nd

Table 1: Ranked comparison of techniques

CCD is the abbreviation given to Cyclic Coordinate Descent, NP to Nonlinear Programming and PFS to Physical Force Simulation.

Computational performance of the three techniques was not directly measured, and a full complexity analysis of the algorithms is left as a topic for further research. In practice, the dynamics-based PFS method appears to have a much higher, more consistent performance than the kinematics-based techniques. A low performance is undesirable as it may lead to low intuitiveness (this was noted in some of the subjective feedback from the user study).

4 Concluding Remarks

We have proposed the use of a skeletal metaphor for graph interaction. Three algorithms were investigated for the provision of this metaphor, and their relative benefits and disadvantages explored.

Much work remains to judge the utility of a skeletal metaphor for graph interaction, particularly applied to the exploration of datasets from specific domains. Large datasets are of interest, as an interaction metaphor combined with an effective layout and clustering may aid in visualising such information.

Research is continuing in this area, and also into the exploration of more complex dynamics-based approaches to implementation of the metaphor. The PFS approach described in this paper suffered significant problems in representing the skeletal metaphor, but these problems may potentially be overcome through more strict constraint satisfaction mechanisms.

5 References

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

Brady, M., Holierbach, J., Johnson, T., Lozano-Perez, T., and Mason, M., editors (1982): Robot Motion: Planning and Control. MIT Press, Cambridge.

Broyden, C. G. (1970): The convergence of a class of double-rank minimization algorithms 2. the new algorithm. J. Inst. Math. Appl. 6, pages 222—231.

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.

Dwyer, T. (2000): Three dimensional uml using force directed layout. Technical Report TR2001/25, Department of Computer Science, The University of Melbourne, Parkville, Australia.

Dwyer, T. (2001): Three dimensional UML using force directed layout. In Eades, P. and Pattison, T., editors, Australian Symposium on Information Visualisation, (invis.au 2001), Sydney, Australia. ACS.

Eades, P., Lai, W., Misue, K. and Sugiyama, K. (1991): Preserving the mental map of a diagram. Technical Report IIAS-RR-91-16E, International Institute for Advanced Study of Social Information Science, Fujitsu Laboratories Ltd.

Fletcher, R. (1970): A new approach to variable metric algorithms. Comput. J. 13, pages 317—322.

Frick, A., Ludwig, A., and Mehldau, H. (1994): A fast adaptive layout algorithm for undirected graphs. In Proceedings of GD'94, volume 894, pages 388—403. Springer-Verlag, Berlin.

Goldfarb, D. (1970): A family of variable metric methods derived by variational means. SIAM J. Appl. Math. 17, 4, pages 739—764.

Herman, I., Marshall, M. S., Melancon, G., Duke, D. J., Delest, M., and Domenger, J.-P. (1999): Skeletal images as visual cues in graph visualization. In Gröller, E., Löffelmann, H., and Ribarsky, W., editors, Data Visualization '99, pages 13—22. Springer-Verlag Wien.

Klein, C. A. and Huang, C.H. (1983): Review of pseudoinverse control for use with kinematically redundant manipulators. IEEE Trans. Syst., Man, and Cybern., vol. SMC-13, no. 2, pages 245—250.

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

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.

Paul, R. P. (1981): Robot Manipulators, Mathematics, Programming and Control. The MIT Press, Cambridge.

Sciavicco, L. and Siciliano, B. (1987): A dynamic solution to the inverse kinematic problem for redundant manipulators. In Proc. IEEE Int. Conf. Robotics and Automation, Raleigh, North Carolina, pages 1081—1087.

Shanno, D. F. (1970): Conditioning of quasinewton methods for function minimization. Math. Comput. 24, pages 647—664.

Walshaw, C. (2000): A multilevel algorithm for force-directed graph drawing. In Proceedings of Graph Drawing 2000, volume 1984, pages 171—182. Springer-Verlag, Berlin.

Wang, L. C. T. and Chen, C. C. (1991): A combined optimization method for solving the inverse kinematics problem of mechanical manipulators. IEEE Transactions on Robotics and Automation, 7(4):489—499.

Ware, C., 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.

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

Welman, C. (1993): Inverse kinematics and geometric constraints for articulated figure manipulation. Master’s thesis, School of Computing Science, Simon Fraser University.

Wolovich, W. and Elliott, H. (1984): A computational technique for inverse kinematics. In Proc. 23rd. Conf. on Decision and Control, Las Vegas, NV.

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.

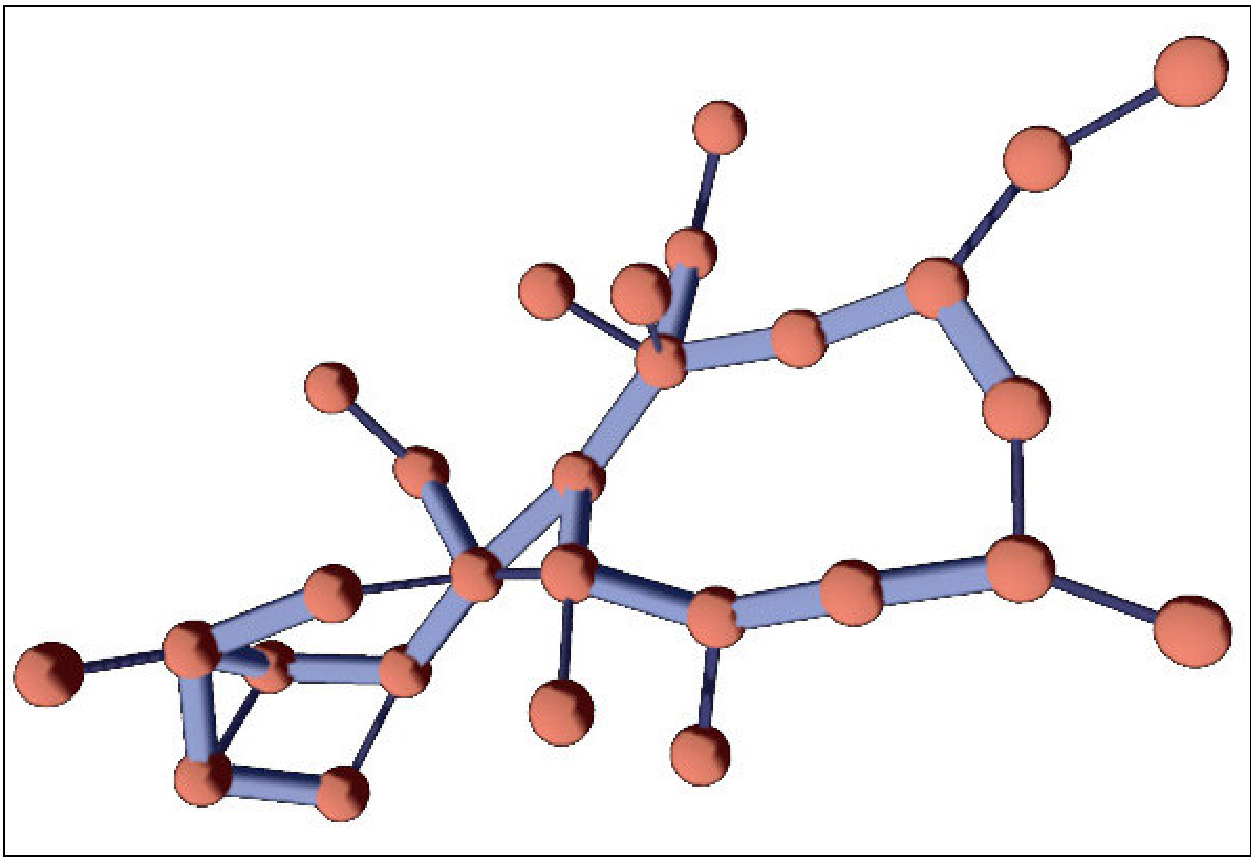

Figure 12: Graph laid out in sGraph

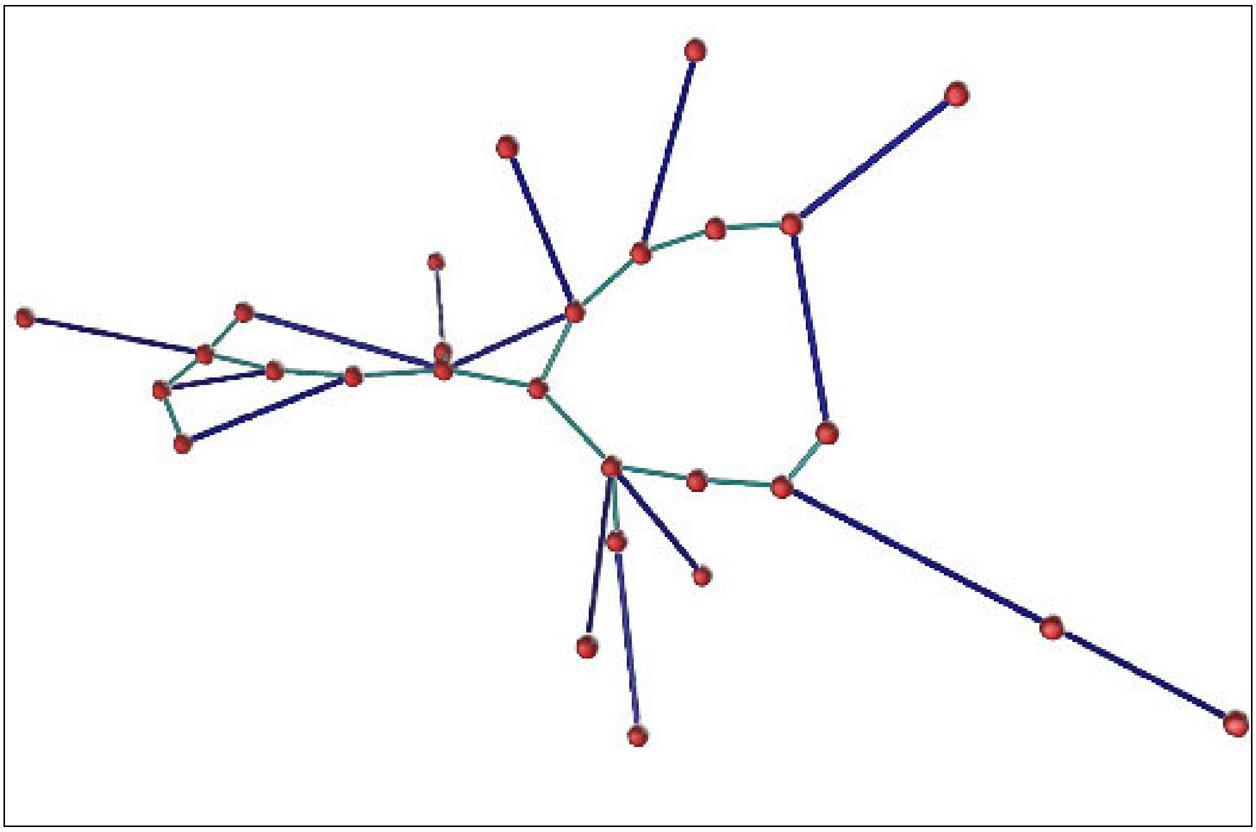

Figure 13: Graph laid out in Wilma