COMAIDE: Information Visualization using Cooperative 3D Diagram Layout

David Dodson

Computer Science Department, City University,
Northampton Square, London EC1V 0HB, UK

dcd@cs.city.ac.uk

Abstract. COMAIDE is a toolkit for user-system cooperation through joint multi-focal graph browsing. It supports cooperative force-directed layout management, concurrent with dialogue handling, for heterogeneous multi-layered 3D interactive diagrams. The layout manager's intuitively 'natural' animations of multi-layered 3D graph drawings support:

1. Cooperative tidying of user-manipulable 3D layout;
2. Optional 'lucid' 3D layout optimized for a favoured viewpoint;
3. Layout annealing for good initial 3D diagram topologies;

1 Introduction

COMAIDE (Co-Operative Multilayer Application-Independent Diagram Environment) is a toolkit for cooperative Diagrammatic User Interfaces (DUIs) [15] to applications written in Prolog. It presents a simple artificial reality of 3D multi-layer node-and-link diagrams supporting multi-focal graph browsing. The nodes of the diagram occupy a set of parallel layers, each node's centre being constrained to lie in the plane of a layer.

COMAIDE diagrams are animated by an innovative force-directed layout manager, dyn, the focus of this paper. As in most force-directed graph drawing systems, dyn simulates repulsion between diagram elements to keep them spread apart, and springiness in links to promote preferred link lengths and alignments.

In dyn's normal, cooperative mode of use, the diagram is animated in a slow and intuitively predictable way which does not interfere with user manipulation of the diagram. This lets the user vary the diagram's 3D topology by dragging nodes. Each node can be dragged within a layer or to another layer by the user. dyn tidies up the result without changing the topology. This is important because the user may see ways of improving the layout that a layout manager cannot find or cannot apply without risk of causing disorientation; also purely automatic layout cannot address unpredictable or ramified user preferences. dyn also has a simulated-annealing mode for producing fresh layouts, e.g. of new diagram contents generated by an application.

Concurrently with layout animation, COMAIDE supports user-application dialogues in which an application responds to mouse actions by modifying diagram content. These dialogues can employ built-in facilities for multi-focal browsing [5, 14, 16, 9] which let both user and application expand, shrink, hide and reveal nodes. This is important in giving the user control of the allocation of screen space (external visual memory) to information whilst letting the application vary this allocation, e.g. to draw attention to important matters. User browsing manipulations incorporate a fast initial layout adjustment in the spatial neighbourhood of the affected node using the 'ductile space' metaphor [5].

Facilities for pop-up and pull-down menus are also built-in. A built-in pull-down menu allows selection of applications and control of the layout manager. Applications can also incorporate specialized layout algorithms for layout initialization as an alternative to the relatively slow but general annealing approach.

COMAIDE is a Prolog environment including sub-systems for application management, menu operations, browsing operations, layout management and diagram I/0, along with various utility sub-systems. A SICStus Prolog process incorporating these sub-systems acts as a client of City's interactive diagram graphics server, icx, which implements ICD-Edit [4, 7, 8] for X/Motif. COMAIDE is currently undergoing further development in the EU CUBIQ project. The version outlined here uses the client-server protocol ADI 1.6.1b detailed in [7].

Section 2 outlines depth perception aspects of COMAIDE diagrams. COMAIDE's layout manager, dyn, is presented and discussed in Sec. 3. Section 4 illustrates both a concept-demonstration of a cooperative Expert System DUI built using COMAIDE and the use of annealing for initial layout. Section 5 concludes.

2 Depth Perception in COMAIDE Diagrams

COMAIDE's graphics server, ICD-Edit, offers a limited 3-dimensionality, '2¾D', in which nodes appear upright and face forwards irrespective of the variable 3D orientation of a diagram. Moreover, ICD-Edit nodes are simply flat rectangles and the links between them are straight lines with optional arrowheads. These constraints allow fast diagram manipulation without 3D hardware but also limit the cues available for depth perception, which thus need special attention.

Naturally, ICD-Edit uses hidden object removal, but this only indicates the sign of the depth difference wherever a node overlaps a node or link,

Perspective is used too. Brookes [1] observes that perspective is very effective as a depth cue when parallel lines and right angles abound in the 3D model. A more detailed analysis for the case of 3D node-and-link diagrams is offered here:

1. Perspective projection is not *in itself* a depth cue.
2. Recognizable spatial arrangements of diagram elements are good depth cues; perspective usually enhances, and in particular disambiguates, this effect.
3. The more parallel lines and right angles there are in the spatial arrangement, the better its depth cueing effect.
4. Significant uniformity of sizes in the model, e.g. amongst nodes or amongst distances between nodes, is needed for perspective to aid depth cueing.

ICD-Edit's optional 'rocking' motion, inspired by SemNet[1], approximates a sinusoidal rotational oscillation with just three co-maintained images. This often seems disturbing at first, but seems generally acceptable. It very usefully depth-cues ICD-Edit diagrams which richly populate 3D space.

COMAIDE diagrams use parallel rectangular layers visualized by 3D bounding boxes and perspective viewing to exploit the potential for depth cueing in ICD-Edit diagrams. Though not shown here, intensity cueing is supported too.

3 The Layout Manager, dyn

As a layout *manager*, dyn runs as a background activity, repeatedly adjusting diagram layout. The successive adjustments approximate the motion of the mutually-repulsive springy diagram elements in a viscous medium. Links are assumed to have negligible mass. Newton's second law applies to each node:

$$\mathbf{F} = m\dot{\mathbf{V}} \tag{1}$$

where m is the mass of the node and $\dot{\mathbf{V}}$ (i.e. $d\mathbf{V}/dt$) is its acceleration vector. The force vector $\mathbf{F}$ is the sum of the motive force on the node and the drag due to viscosity. Using a simple model of viscosity, neglecting node size, this gives:

$$m\dot{\mathbf{V}} = \mathbf{F}_{\text{MOTIVE}} - k\mathbf{V} \tag{2}$$

where k is a viscosity coefficient. The option of treating nodes as having negligible mass is usually taken, so that the motion equation degenerates to:

$$\mathbf{V} = \mathbf{F}_{\text{MOTIVE}}/k \tag{3}$$

In this case, in each time step, movement is proportional to applied force — a common feature of force-directed graph drawing following its use in [10].

Fig. 1 summarizes dyn's layout animation algorithm. Table 1 lists control parameters which can be varied using pull-down menus while dyn is running.

3.1 Motive Forces in dyn

Nine different types of motive-force interaction are considered, as follows.

3D Node-boundary repulsion: Each node is repelled inwards by each face of a model-axis-aligned cuboid [6] by a force increasing linearly from zero at the cuboid's centre to 1 at a distance of bdry_rim_width from the boundary and then to max_repulsion for a node touching, crossing or outside the boundary.

3D Node-node repulsion: Each node pair within a layer undergoes inverse square law repulsion subject to a force limit of max_repulsion and a distance limit of repel_horizon.

3D Node-link repulsion: For each node, for each link which is not a link of that node and which either lies in the same layer as the node or crosses that

[1] SemNet [11] demonstrated fully 3D, mainly viewpoint-navigated knowledge browsing with scale-driven recursive decomposition of spatial cluster nodes)

```
For each node n: Set V(n) {the velocity of n} to [0,0,0];
While ts>0 {where ts is the assumed size of time step in seconds}:
   <<< FORCE Computation: >>>
   For each node n: Set F(n) {the motive force on n} to [0,0,0];
   For each node n: Add its boundary repulsion forces to F(n), also
   For each force couple between two diagram elements:
      For each member E of the pair of elements:
         If E is a node n: Add the relevant force to F(n)
         else (E is a link from node n1 to node n2):
            Add the relevant forces to F(n1) and F(n2);
   <<< MOTION Computation: >>>
   If inertia>0 then:
      tss := ts/5;
      For each node n, repeat 5 times:
            Posn(n) := Posn(n) + V(n)*tss;
            V(n) := V(n) + (F(n)-V(n)*viscosity)*tss/(mass(n)*inertia);
   else For each node n:
            V(n) := F(n)/viscosity;
            Posn(n) := Posn(n) + V(n)*ts;
   viscosity := min( end_viscosity, viscosity*(100+anneal_rate)/100 );
```

Fig. 1. Outline of dyn's layout management algorithm

Parameter	Description
bdry_rim_width	Distance at which node-boundary repulsion is 1
n_n_3_repel_factor	Coefficient of 3D node-node repulsion
n_l_3_repel_factor	Coefficient of 3D node-link repulsion
l_l_3_repel_factor	Coefficient of 3D link-link repulsion
repel_horizon	Distance limit of repulsion between diagram elements
max_repulsion	Limit on each 3D repulsion force
perpendicity	Stiffness coefficient of between-layer links
preferred_length	Ideal forward size of in-layer links
fwd_rigidity	Forward stiffness coefficient of in-layer links
planar_field	Stiffness coefficient of 'magnetic' link alignment
n_n_2_repel_factor	Coefficient of 2D node-node repulsion
n_l_2_repel_factor	Coefficient of 2D node-link repulsion
max_2d_repulsion	Limit on each 2D repulsion force
viscosity	Ratio of drag force to velocity (negated)
anneal_rate	% increase in viscosity per layout step
end_viscosity	Limit to increase of viscosity
inertia	Ratio of inertia to mass

Table 1. dyn's Control Parameters

layer, inverse square law repulsion occurs between them subject to a force limit of max_repulsion and a distance limit of repel_horizon.

3D Link-link repulsion: Each pair of links such that one is in a layer which the other crosses or both cross the same gap between layers experience inverse square law repulsion acting along the line of shortest distance between the links, subject to a force limit of max_repulsion and a distance limit of repel_horizon, provided they would not be closer if extended to infinity.

Between-layer link alignment stiffness: Each link from a node in one layer to a node in another exerts a pair of forces, one on each node, of magnitude perpendicity*tan(angle from the perpendicular), seeking to make the link perpendicular to the layers. The effect is analogous to barycentering [17], but generalized to 3D.

In-layer link alignment stiffness: Each link in a layer with a forward (i.e. upward) direction is considered to be 'magnetic' if planar_field exceeds zero.

A non-magnetic link exerts a force pair seeking, with a linear elastic stiffness of fwd_rigidity, to make it preferred_length long in its current orientation.

A 'magnetic' link exerts a force pair seeking alignment in the layer's forward direction, D. The force components in D are zero if the length component in D is preferred_length, and otherwise reflect a linear elastic stiffness of fwd_rigidity. The force components in the perpendicular direction S within the layer are zero if the length component in S is zero, and otherwise reflect a stiffness proportional to planar_field. This crude but fast approximation of magnetism seems to avoid instability and to have intuitive appeal and a barycentering-like effect [17].

2D Node-node and node-link repulsion: These weak 2D repulsions are somewhat similar to their 3D counterparts, but arise in the display plane, ignoring depth differences between nodes. Their effects are, respectively, to minimise node overlap and to resist node-link crossings in the 2D display.

3.2 Force, Mass and Heat in Nature and in dyn

COMAIDE animates the diagram approximately as if suspended in a thick fluid which continually promotes equilibrium by extracting energy from it. Low-energy equilibrium states in force-directed layout have no kinetic energy because they are static, and have low potential energy in force interactions, implying good satisfaction of the separation and alignment preferences modelled by the force interactions, averaged over the diagram. The motion and heat that would be induced in a real fluid is ignored, but compared to most force-directed graph drawing systems, e.g. [13, 2, 12], dyn's dynamics are relatively natural.

dyn's universal inertial constant, inertia, controls the magnitude of the effect of mass, i.e. delay in response of motion to force and tendency to overshoot. When inertia is zero, diagram layout typically seems to converge faster with similar or perhaps better results, though *no* formal comparison has been made. Moreover, the animation seems more open to intuitive prediction when mass effects do not need to be considered. Indeed, naturally speaking, the more viscosity dominates mass (i.e. the smaller velocities are than they would be in the absence of viscosity), the smaller the effect of mass on trajectory.

In practice, in the course of a time step, diagram elements tend to approach each other more closely than their initial energy would allow in reality. This happens because the crude simulation algorithm does not foresee the escalation of repulsion between them in the course of their approach. Compared with a hypothetical perfectly natural animation the effect is to provide a source of random mechanical energy. A positive viscosity is needed just to counterbalance this effect. Adopting the analogy between diagram structure and molecular structure, an increase in energy equates to heating. Thus a low viscosity in dyn tends to produce heating, whilst a high viscosity tends to produce cooling.

3.3 Normal and Annealing Modes of Layout Management

dyn's 'normal' mode of cooperative operation occurs when diagram energy is low enough for diagram elements to succeed in repelling each other. In other words, between one force computation and the next, no diagram element (node or link) passes through another due to motion so fast that their high mutual repulsion when close together is not sampled — or if sampled, is insufficient to prevent through-passing. This equates to the ordinary reality of solid objects. In some sense, the 3D topology of the diagram remains constant.

At high energy levels, the diagram moves chaotically with lots of through-passing. This is somewhat analogous to the behaviour of fluids, particularly as the positions of nodes are much less constrained by the links between them.

The term annealing traditionally meant the slow cooling of a metal such that the boundaries between its crystals have time to migrate, relative to its atoms, to a particularly low energy state associated with large crystal size and with high ductility. Through metaphor, it can be now used for analogous slow reductions of energy which tend to yield particularly low-energy equilibrium states. Dropping the qualification 'simulated' from annealing seems particularly valid when, as in graph drawing, reality is being created more than simulated.

Unlike many simulated annealing systems, such as [2], dyn is deterministic, being controlled by increasing viscosity rather than by explicitly reducing the probability of random energy increase. Annealing is used to solidify the diagram into a low-energy 3D topology. The diagram can initially be melted with low viscosity and/or by 'crushing' it, which positions each node at the centre of its layer, creating high initial potential energy. If the diagram 'explodes' excessively, 'sandbags' can be switched on to limit node positions to a cuboid model space.

3.4 3D and 2D Layout

Diagram annealing in dyn using 3D forces leads to fairly good 3D layouts, although some cooperative user input is often helpful once the diagram is solid. These layouts are not however optimized for a particular direction of viewing. To maintain familiar visual context and aid fast recognition of familiar content, the diagram needs to be arranged appropriately for some principal viewing direction. dyn achieves this using 2D node-node and node-link repulsions which arise in the 2D display projection of the diagram. By making these 2D repulsions small compared to the 3D repulsions, they minimize node-node and node-link overlap in the projected view, without significantly impairing the quality of the

3D layout. The resulting layout is good both in 3D and in the 2D-displayed view. The 2D forces also tend to maintain the 2D topology of a solid diagram and are not useful to consider until the diagram is solid, especially as they currently take longer to compute than the 3D ones.

Gas, liquid and solid are familiar thermodynamic phases of everyday substances. VIM diagrams have three thermodynamic phases too: the *fluid* phase, the ordinary *solid* phase, and a third phase obtained with added 2D repulsions which can add perspicuity to solid diagrams. This third phase is the *lucid* phase.

Clearly there is much more to be gained from the correspondence between materials science and 3D force-directed graph drawing than is uncovered here.

4 Sample Results

Note that when COMAIDE presents the diagrams shown below, ICD-Edit's rocking motion makes them look far more 3-dimensional. This makes them look less tangled and resolves structural ambiguities.

4.1 Cooperative Browsing Supported by Cooperative Layout

This example shows VIM [9], a visual expert system prototype, running in COMAIDE. The underlying expert system shell is IM1 [3].

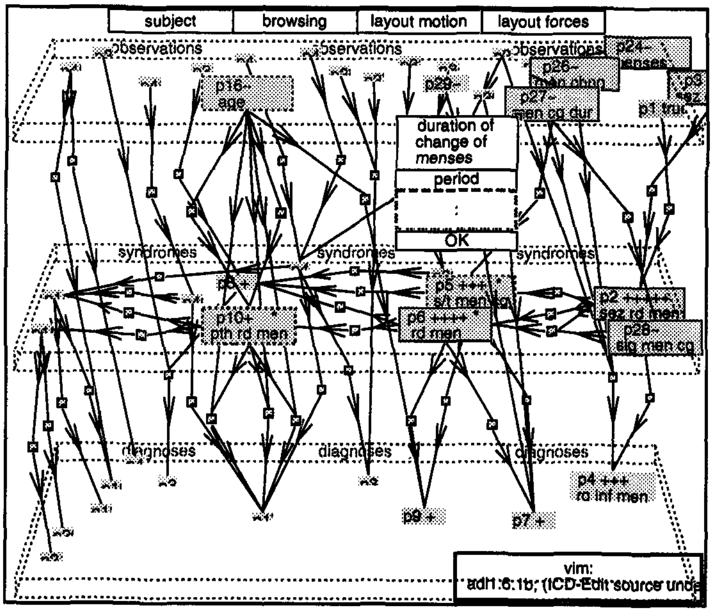

Fig. 2. VIM on first selecting a question to be answered

Figure 2 shows a view of a 3D overview diagram comprising 89 nodes and c.100 links. It shows a small IM1 knowledge base of 42 particulars (the oblong nodes) and 47 rules (the square nodes and their links). The user has entered, as the initial input, the proposition that the patient complains of amenorrhoea (particular p1, top right front). As a result, IM1 has updated various weights of evidence (visually encoded mainly with '+' and '−' characters) by forward rule application (i.e. with the data flow shown by arrowheads). It has then also updated 'investigative importances' (visually encoded by oblong-node border density) by reverse rule application. VIM has used these updates to estimate the pertinence (i.e. desired salience) of the affected nodes and has updated the diagram accordingly. dyn has then executed two cycles of layout adjustment, chiefly spreading out enlarged nodes to reduce overlap. Finally the user has clicked on the 'men cg dur' node causing a data entry pop-up to appear.

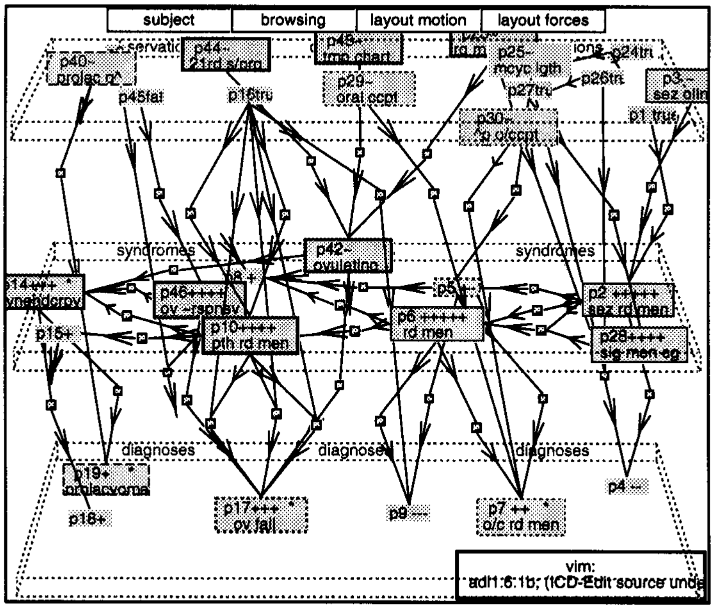

Fig. 3. Later in the same VIM consultation

Figure 3 shows a later stage in the consultation. The layout manager has executed a further 12 layout adjustment cycles. Concurrently, the user has:

1. Provided 3 answers to questions;
2. Rearranged three nodes to release a snag which inhibited layout improvement, without altering the 3D topology of the diagram;
3. De-selected most of the 'tiny' particulars from view (Ideally VIM would keep impertinent particulars hidden, but at the time of writing its automatic browsing actions vary node salience but not node selection.)

Node de-selection can be reversed by clicking on the 'link stubs' which point in the direction of omitted detail. Each such click reveals a linked rule node, together with any other links from that rule to particulars shown in the diagram.

dyn currently takes almost two minutes per animation step for the full lucid VIM diagram shown in Fig. 2, using lists for vectors in SICStus Prolog on a Sun SPARCclassic. Software tuning and quadrupled hardware power are expected to yield an acceptable speed of 1 to 5 seconds per cycle for such a diagram.

4.2 Cooperative Annealing

Fig. 4 shows a diagram which has been crushed and then given 29 annealing steps with an initial viscosity of 0.2, rising 2% per step. All of the other parameters have their default values, with 2D forces off. The diagram solidified within 11 annealing steps, and moved very little after 20 steps. Each layout step took about 2 seconds, including Prolog, ICD-Edit and X server execution time.

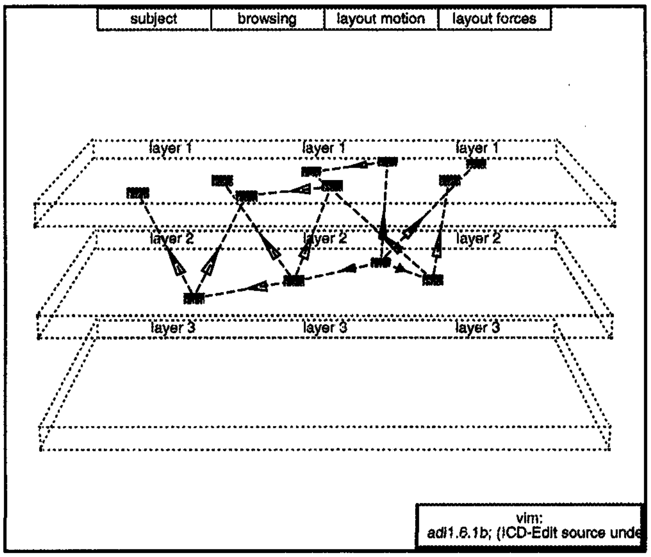

Fig. 4. A Diagram after 29 Annealing Steps

After repositioning one node by simple dragging, 10 more annealing steps produced the layout shown in Fig. 5, in which the dragged node is emphasized. dyn was stopped to aid this, as node dragging was not suppressing repositioning by dyn at that time, making dragging prone to failure in small diagrams.

The right hand half of the diagram was then manually repositioned to make the diagram *roughly* symmetrical, again with dyn turned off. After about ten

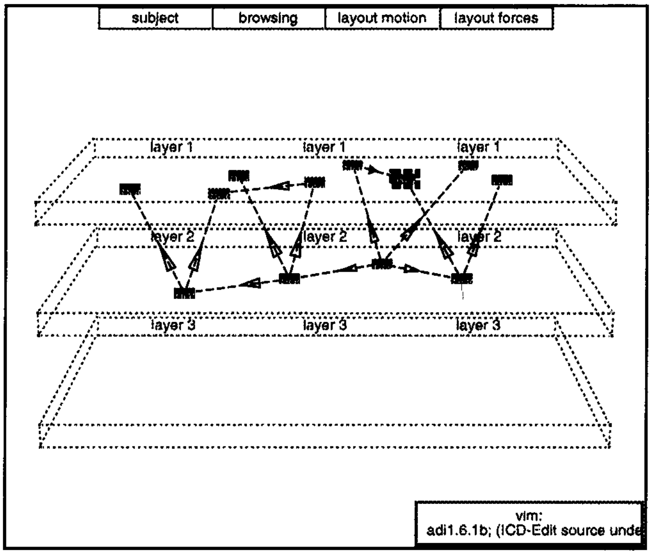

Fig. 5. After 39 Steps

more layout cycles, the node velocities were below 1 pixel per cycle. Fig. 6 shows the layout a minute later. A further 250 steps produced no noticeable result except for a very slight clockwise rotation as viewed from above.

5 Conclusion

COMAIDE demonstrates a lot of support for user-application cooperation in joint multi-focal graph browsing, illustrating key benefits of 3D in this context. The key ingredient is a cooperative layout manager, dyn, which animates multi-layered 3D graph drawings in a fairly graceful, subjectively natural and intuitively comprehensible way. Animation by dyn supports:

1. Cooperative management of 3D layout, in tidying layouts without changing their topology while allowing user manipulation;
2. Optional combined 3D and 2D 'lucid' layout for a favoured viewpoint;
3. Layout annealing, finding good 3D diagram topologies.

dyn's novelty rests largely on its rich model of diagram structure and forces:

1. Both sorts of diagram element (links as well as nodes) repel each other.
2. Links within planes are treated distincity from links between planes.
3. An optional pseudo-magnetic force promoting link alignment within a plane is modelled as an orthogonal pair of simple spring forces.
4. 2D forces which arise in the display plane are used to gently coerce good 3D layouts into being good 2D ones from the chosen viewpoint.

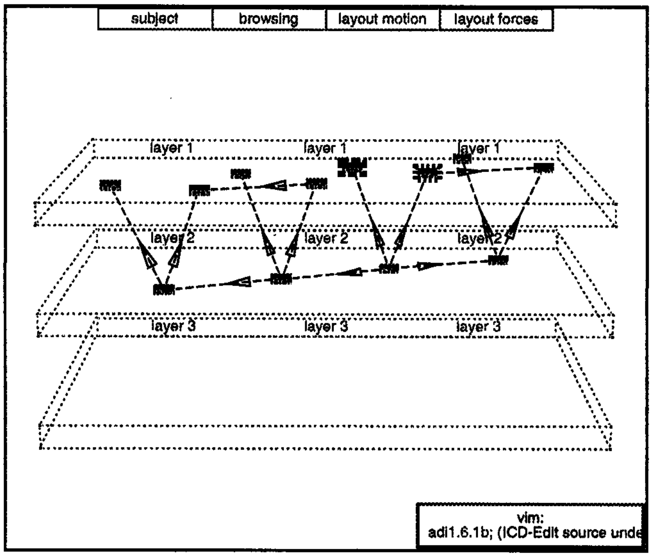

Fig. 6. After another 5 Drags and 40 Steps

5. The force computation ignores hidden diagram elements and the motion computation ignores any component of force on a node normal to its layer.

Further details including references to related work can be found using the World Wide Web URL: http://web.cs.city.ac.uk/research/dig/digpapers.html

6 Acknowledgements

Funding by EPSRC (GR/G56478, Cooperative KBS Browing), EU COPERNICUS (project 10979, CUBIQ) and City University is gratefully acknowledged.

References

1. FP Brooks. Grasping Reality Through Illusion - Interactive Graphics Serving Science. In SIGCHI Bulletin (special issue), ACM/Addison Wesley. (*CHI'88 Proceedings*, Washington, May 1988).
2. R Davidson and D Harel. Drawing Graphs Nicely Using Simulated Annealing. Tech. Report, The Weizmann Institute of Science, Rehovot, Israel, 1989; revised 1992,1993. To appear in Communications of the ACM.
3. DC Dodson and AL Rector. Importance-driven control of diagnostic reasoning. In MA Bramer, editor, *Research and Development in Expert Systems*. Cambridge University Press, 1984. (Proceedings of Expert Systems 84, Warwick, December 1984).

4. DC Dodson, LH Reeves and RB Scott. ICD-EDIT: - A Server for $2\frac{3}{4}$D Interactive Connection Diagram Graphics with Prolog Clients. Technical report TCU/CS/1995/1, Dept. of Computer Science, City University. 9pp. (Poster, Graph Drawing 94, 10-12 October 1994, Princeton, New Jersey). World-Wide Web (colour): http://web.cs.city.ac.uk/research/dig/95/TCU_1995_1/p1.html
5. DC Dodson. TRIVIAL: Refocusing in Cooperative Diagrams with Ductile Space. Tech. report TCU/CS/1995/2, Computer Science Dept., City University. 10pp. (Poster, Graph Drawing 94, 10-12 October 1994, Princeton, New Jersey). WWW (colour): http://web.cs.city.ac.uk/research/dig/95/TCU_1995_2/p2.html
6. DC Dodson. N-Dimensional RSPs, Right Multilayered Diagrams and Prolog. Technical report TCU/CS/1995/3, Dept. of Computer Science, City University. 9pp. (Poster, Graph Drawing 94, 10-12 October 1994, Princeton, New Jersey). WWW (colour): http://web.cs.city.ac.uk/research/dig/95/TCU_1995_3/p3.html
7. DC Dodson, LH Reeves and RB Scott (1995) ICD-Edit's Client-Server protocol: ADI Version 1.6.1. Technical report TCU/CS/1995/x, Dept. of Computer Science, City University, 1995. World-Wide Web: http://web.cs.city.ac.uk /informatics/cs/research/dig/95/adi1_6_1/sp.ps
8. WWW ICD-Edit page: http://web.cs.city.ac.uk/research/dig/icd-edit.html
9. DC Dodson, JA Secker, RB Scott and LH Reeves. VIM: 3D Co-operative Diagrams as KBS Surfaces. To appear in MA Bramer, editor, *Research and Development in Expert Systems XII.* (Proc. Expert Systems 95, Cambridge, Dec. 1985). WWW (colour): http://web.cs.city.ac.uk/research/dig/95/es95/es95.ps (Oct 95).
10. P Eades. A Heuristic for Graph Drawing. *Congressus Numerantium*, 42:149–160, May 1984.
11. KM Fairchild, SE Poltrock and GW Furnas. SemNet: Three Dimensional Graphic Representations of Large Knowledge Bases. In R Guidon, editor, *Cognitive Science and its Application for Human Computer Interaction.* Lawrence Erlbaum, 1988.
12. A Frick, A Ludwig and H Mehldau. A Fast Adaptive Layout Algorithm for Undirected Graphs. In R Tamassia and IG Tollis, editors, *Graph Drawing.* Lecture Notes in Computer Science 894, Springer, 1995. (Proc. DIMACS International Workshop, GD94, Princeton, New Jersey, USA, October 1994).
13. T Fruchterman and E Reingold. Graph Drawing by Force-Directed Placement. *Software-Practice and Experience*, 21(11):1129–1164, 1991.
14. K Kaugars, J Reinfelds and A Brazma. A Simple Algorithm for Drawing Large Graphs on Small Screens. In R Tamassia and IG Tollis, editors, *Graph Drawing.* Lecture Notes in Computer Science 894, Springer, 1995. (Proc. DIMACS International Workshop, GD94, Princeton, New Jersey, USA, October 1994).
15. T Lin and P Eades. Integration of Declarative and Algorithmic Approaches for Layout Construction. In R Tamassia and IG Tollis, editors, *Graph Drawing.* Lecture Notes in Computer Science 894, Springer, 1995. (Proc. DIMACS International Workshop, GD94, Princeton, New Jersey, USA, October 1994).
16. EG Noik. Encoding Presentation Emphasis Algorithms for Graphs. In R Tamassia and IG Tollis, editors, *Graph Drawing.* Lecture Notes in Computer Science 894, Springer, 1995. (Proc. DIMACS International Workshop, GD94, Princeton, New Jersey, USA, Oct. 1994).
17. K Sugiyama, S Tagawa and M Toda. Methods for Visual Understanding of Hierarchical System Structures. *IEEE Trans. Systems, Man and Cybernetics*, 11(2):109–125, February 1981.