# An Interactive Constraint-Based System for Drawing Graphs


| Kathy Ryall | Joe Marks | Stuart Shieber |
|---|---|---|
| Thornton Hall | MERL | Engineering Sciences Lab |
| University of Virginia | 201 Broadway | Harvard University |
| Charlottesville, VA 22903 | Cambridge, MA 02139 | Cambridge, MA 02138 |
| +1-804-982-2200 | +1-617-621-7534 | +1-617-495-2344 |
| ryall@cs.virginia.edu | marks@merl.com | shieber@eecs.harvard.edu |



### ABSTRACT

The GLIDE system is an interactive constraint-based editor for drawing small- and medium-sized graphs (50 nodes or fewer) that organizes the interaction in a more collaborative manner than in previous systems. Its distinguishing features are a vocabulary of specialized constraints for graph drawing, and a simple constraint-satisfaction mechanism that allows the user to manipulate the drawing while the constraints are active. These features result in a graph-drawing editor that is superior in many ways to those based on more general and powerful constraint-satisfaction methods.



### KEYWORDS

Graph drawing, constraint-based layout, drawing tools, collaborative interfaces.


### OVERVIEW

The dominant metaphor in the design of human-computer interaction is the view of computer as *servitor*. Although interfaces based on this view have proved useful in very many settings, qualitative progress in the area of human-computer interaction may await the ability to interact with the computer as *collaborator*. We have built a system, called "GLIDE" (Graph Layout Interactive Diagram Editor), for interactive graph layout,[1] that organizes the interaction in a more collaborative manner than in previous graph-drawing systems. The GLIDE system is a constraint-based drawing editor designed specifically to enable users to easily produce small and medium-sized diagrams. The user is responsible for an approximate layout of the nodes, and for specifying declaratively the overall visual organization of the diagram. The computer converts these visual-organization requirements into a set of constraints expressed in terms of a simple physical model, and calculates the positions of graphical objects via physical simulation. Unlike traditional reactive interfaces in which the user makes a request and waits for a response, GLIDE is a fully interactive system in which both user and computer can affect the state of the drawing simultaneously. The system does not pause while the user adds or moves nodes and edges, or applies or deletes constraints. Likewise, the user can intervene and continue to interact with the drawing as the system adjusts node positions in an attempt to satisfy the various user-specified constraints.

More specifically, GLIDE incorporates a carefully chosen set of "macro" constraints, or Visual Organization Features (VOFs), which are listed in Figure 1; the application of each VOF is illustrated by *before* and *after* layouts.[2] The user-specified VOFs are applied by converting them into a set of spring-like forces affecting the nodes in a graph drawing. Additional forces are introduced automatically to preserve syntactic correctness of the drawing. The user may also apply force directly to a node by dragging it with the mouse. The nodes, which are modeled as point masses, are moved into minimum-energy configurations by a physical simulation based on a generalized force-directed algorithm [5], whose behavior is easily appreciated and influenced by the user. The simulation of the physical model is animated continuously, thus providing useful visual feedback to the user [1, 16]. Although a weak mechanism for satisfying constraints, energy minimization through physical simulation handles over-constrained systems gracefully, and provides an easily understood metaphor for the user.

[1] By "graph" we mean the node-edge network diagrams widely used to visualize binary relations, and not arbitrary informational graphics. There is a considerable literature on graph-drawing algorithms, but it primarily concerns noninteractive techniques for automatic graph layout [2, 3, 11, 12, 15].

[2] These VOFs have been incorporated previously in two batch systems for graph layout [4, 9]. The present system is the first to allow interactive specification and manipulation of persistent high-level VOFs such as these. In related work, Henry and Hudson have described an interactive graph-layout system that uses a different set of nonpersistent layout operators [7].

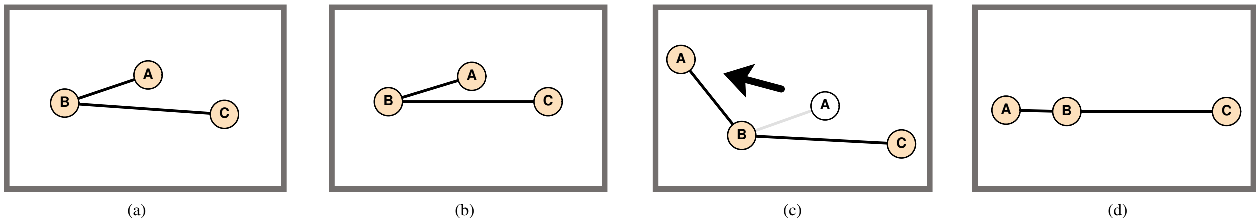


Figure 2: User intervention is required to find a globally optimal solution.

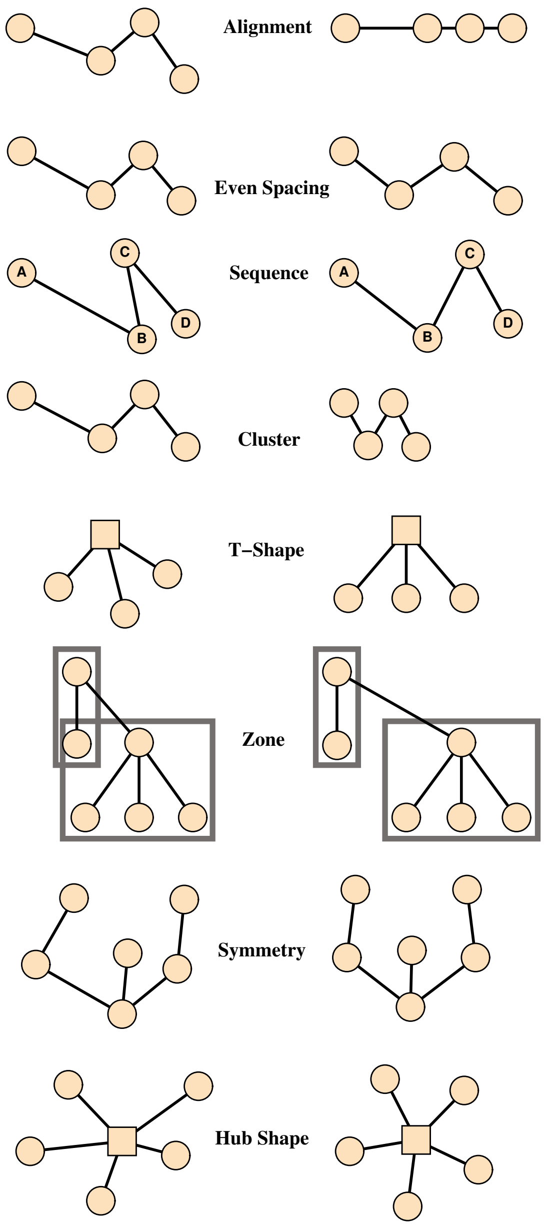


Figure 1: Visual Organization Features (after [9]).

In GLIDE, user manipulations are useful not only for exploring design alternatives but for providing "advice" to the system when it finds itself in a local optimum. Figure 2 is a simple example. In Figure 2(a), we see three nodes, connected by two edges. The user has added a single Alignment VOF to the set of nodes. The system attempts to satisfy this constraint by moving the three nodes toward an implicit horizontal line running through the vertical centroid of the three nodes; meanwhile, the syntactic constraint prohibiting overlap provides a repulsive force between nodes and edges. In Figure 2(b), GLIDE has moved node A down, and nodes B and C upwards. The three nodes cannot be aligned, however, due to the edge between B and C. By manually moving A anywhere to the left of B (Figure 2(c)), the user obtains an optimal arrangement (Figure 2(d)).

These interactions occur as the simulation is running, and allow the computer and system together to collaborate in finding better global solutions to the implicit constraint-satisfaction problem. Such advice is especially useful in cases of over- or underconstrained designs. If a set of VOFs generates an overconstrained layout, GLIDE will find the nearest stable configuration, which may satisfy different VOFs to varying degrees. The user, who controls the design process at a global level (the choice of VOFs and the gross placement of nodes in the diagram), can easily guide the computer to find more satisfactory solutions and acceptable layouts by moving nodes or adjusting VOFs. In case of underconstrained layouts with multiple solutions, the user can also provide advice to explore alternatives.

In the next section we illustrate the use of GLIDE by describing a typical interaction from a user's perspective. We then describe the underlying constraint-specification and -satisfaction mechanisms in detail.

## EXAMPLE INTERACTION

Figures 3 to 7 show snapshots of various intermediate stages in the process of drawing a given graph. Figure 3 depicts the complete system interface; other figures show only the canvas area.

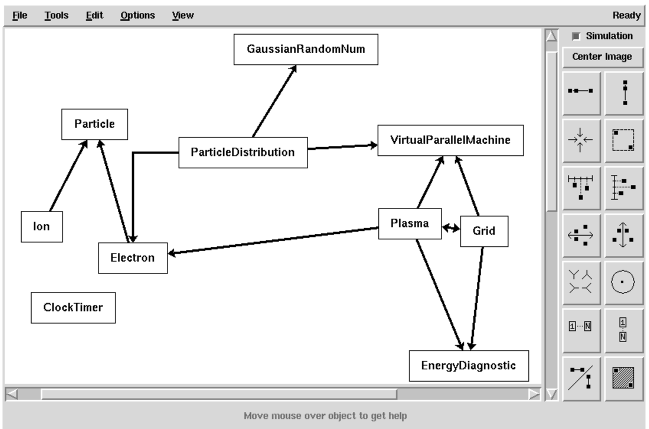


Figure 3: Initial layout, with labeled nodes and edges, shown in the context of the GLIDE interface.

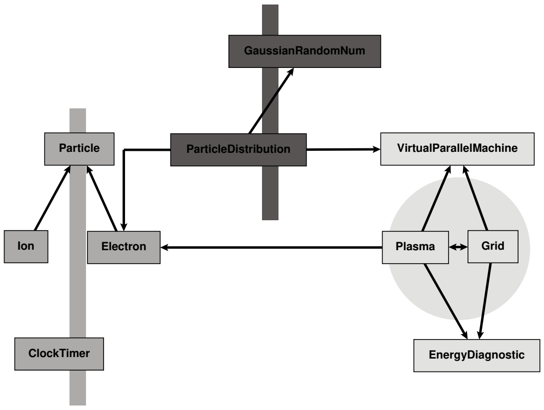


Figure 5: User adds three more VOFs: *Symmetry, Alignment* and *Hub Shape*.

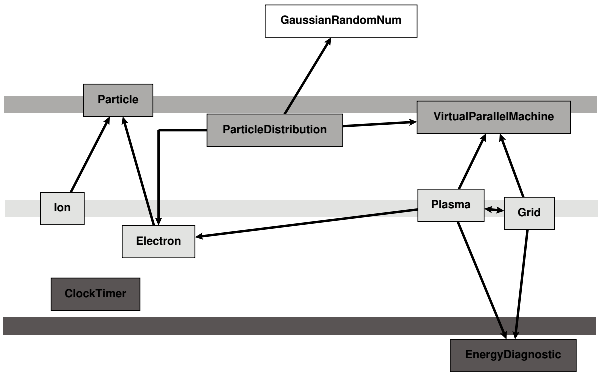


Figure 4: User adds an *Alignment* VOF to each row.

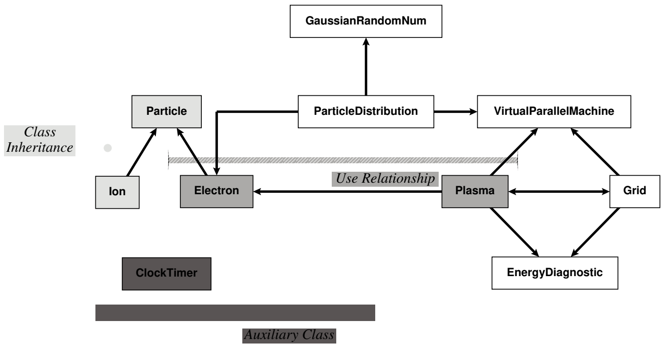


Figure 6: User adds three text labels, and three VOFs: *Clustering, Even Spacing* and *Alignment*.

For simplicity of exposition, the example interaction depicted here is based on a task in which the user attempts to create a particular layout already envisioned, rather than exploring alternative layouts. The user places a set of nodes in approximately the desired layout using standard direct-manipulation techniques. Even at this stage, however, the system is helping out by automatically enforcing prohibitions of overlapping nodes; nodes too close together will be repelled from each other.[3]

Edges, directed or undirected, linear or orthogonal, can be added between nodes. Figure 3 shows the layout after an initial node- and edge-placement phase. Note the orthogonal edge between the "ParticleDistribution" and "Electron" nodes. The system will maintain the orthogonality constraint (comprising two alignment constraints and a phantom node – see below) throughout the design process.

To add a VOF to the layout, the user first selects a set of nodes using standard mouse techniques such as clicking or region-dragging. The user may then apply one or more VOFs to the set by pressing the appropriate push buttons, located on the right of the window in Figure 3. In Figure 4, the user has applied a horizontal Alignment VOF to the second, third, and fourth rows of nodes. As VOFs are applied to the nascent diagram, graphical indicators of the constraints are added. In the implemented system, the graphical VOF indicators are shown visually by a dynamic highlighting mechanism that cannot be replicated in static images. We therefore use shades of gray to indicate different VOFs in the figure; the grey rectangles in Figure 4 serve as a graphical indicator of the Alignment VOFs.

[3]As is standard in drawing tools, the user has control over a variety of graphical properties of the nodes, including shape, font, background color, foreground color, border color, and dimensions. These are derived from system defaults, which can be set by the user, and modified at any time via a dialog box. Each node is automatically sized to accommodate its label; if a node's label is changed, the system will automatically enlarge a node to accommodate its new label. These capabilities are not shown in the figures.

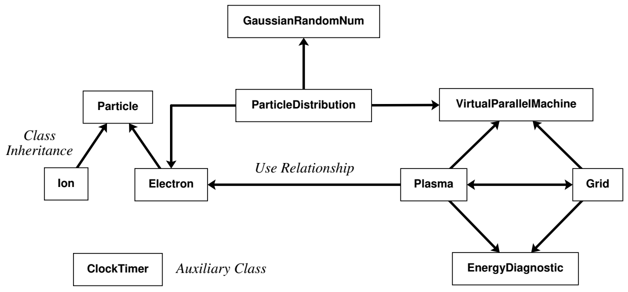


Figure 7: The finished layout.

The system satisfies these constraints via the simulation algorithm described in more detail below. Indeed, movement of the nodes to satisfy the constraints would typically proceed while the user is adding more VOFs.[4] Figure 5 shows the stable configuration that ensues after the three Alignment VOFs have been satisfied, along with three more VOFs the user has added. On the right, the user has added a Hub Shape VOF, indicated as a light gray circle. The center two nodes (dark gray) are to be vertically aligned. Finally, the four nodes on the left have had a Symmetry VOF applied to them.

Once again, the system attempts to satisfy all of the constraints, both old and new, in determining each node's placement. Figure 6 shows the updated node positions. Each row is still aligned, and the new VOFs have been satisfied as well. In addition, the user adds three text labels to the layout. Text labels are simply a specialized node type. The user has applied three more VOFs, which will better position each of the text labels. The light-gray nodes on the left are subject to a Cluster VOF. The dark-gray nodes on the bottom are to be horizontally aligned. The middle three nodes, shaded medium gray, are to be evenly spaced. Figure 7 shows the final layout generated by the full set of nine VOFs.

## IMPLEMENTATION

From a user's perspective, GLIDE is a simple high-level interface for adding and deleting nodes and edges, and for applying and removing various VOFs, thereby inducing new drawings. This facade is maintained by the underlying system, which is continually translating user actions into low-level constraints that it then tries to satisfy. In this section, we describe the relationship between the high-level VOFs and the low-level constraint mechanisms, and how the constraints are satisfied by physical simulation. In addition, we also describe how the constraints and constraint-satisfaction process are made apparent to the user.

### Constraint Formulation

The fundamental low-level constraint mechanism is essentially a spring that obeys Hooke's Law; graph nodes move according to the spring-like forces attached to them.[5] A mass-spring model for graph drawing was first proposed by Eades [5], but in his and most subsequent systems, the spring forces correspond to topological or geometric properties of the graph. In the GLIDE system (as in [4]), a more general notion of spring force is used.

Constraints on graphs fall into two main classes, syntactic and semantic. Syntactic constraints are universal requirements necessary for a diagram to be well-formed. The GLIDE system respects two such constraints (see Figure 8):

- *Node-node overlap:* Two nodes should not overlap. This constraint is enforced by placing a spring between each pair of nodes. The spring's rest length is the required minimum distance between nodes, but it also has the property that its spring constant reduces to zero when stretched beyond its rest length. The spring therefore only applies force when the two nodes overlap, compressing the spring. Overlapping is thus prevented, but movement apart is not penalized. This is one of several ways in which the simulation is rendered nonphysical by generalizing the notion of a spring.

- *Node-edge overlap:* A similar spring is placed between each node-edge pair. Nodes are the only objects to which force can be applied, so the goal of applying a force to an edge is actually accomplished by applying half the force to each of the two nodes at the edge's endpoints. This disassociation between the spring's conceptual endpoints and the points of application of the spring's force is another example of how our model differs from more physically faithful mass-spring systems.

Applying these two syntactic constraints alone, the system enforces the well-formedness of diagrams. Semantic constraints, expressed as VOFs, enhance the visual form and communicative power of the drawing. GLIDE supports the following VOFs, implemented with sets of generalized springs as described (see Figure 8):

- *Alignment (horizontal, vertical, either):* The set of nodes should be collinear and axis-aligned. A spring with rest length zero is attached between each node

[4] The GLIDE system is written in Tcl/Tk, with the simulation written in C. The simulation runs continually, polling for user input using the `update` facility during screen updates. In this manner, user input can be immediately reflected in the physical simulation.

[5] Hooke's Law states that strain, the ratio of the change in length to the original length, is proportional to the stress that produces it. (*"Ut tensio, sic vis."*) In addition to spring-like forces, all nodes are subject to a global damping force for stability.

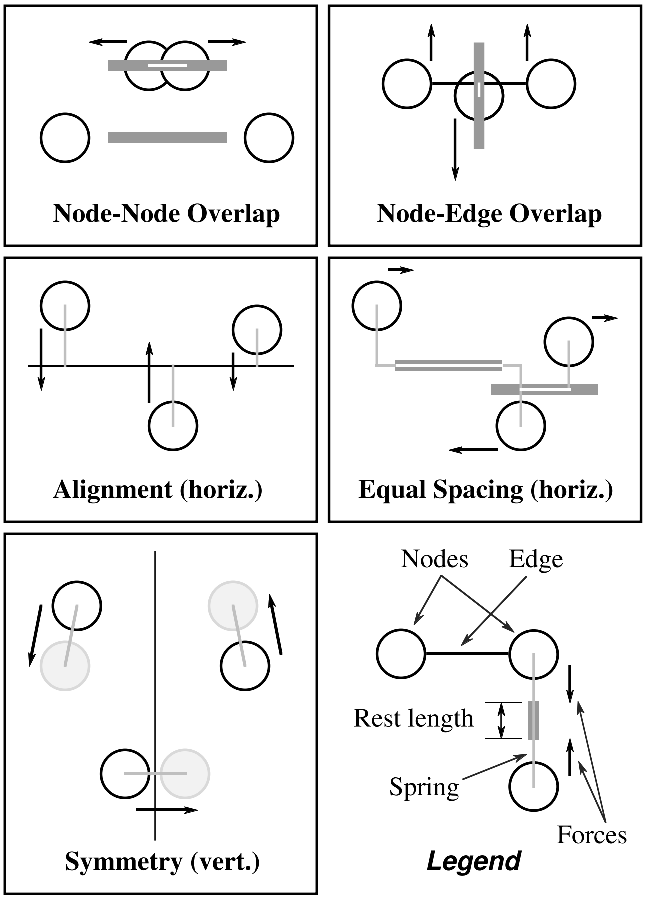


Figure 8: Reduction of some sample VOFs to systems of generalized springs.

and a virtual axis-aligned line through the centroid of the nodes. Note that forces are applied only to the nodes, and not to the virtual axis. In the case of a horizontal or vertical Alignment VOF, the axis alignment is to the respective axis. The third case uses the axis to which the nodes are already most closely aligned.

- *Equal Spacing (horizontal, vertical):* The nodes should be spaced evenly along the given axis. Spacing along the orthogonal axis is unconstrained. Adjacent pairs of nodes are connected with springs whose rest length is the computed average distance between adjacent nodes.

- *Sequence (horizontal, vertical):* The nodes should be ordered in the current sequence along the given axis. Springs with a very short rest length and with asymmetric spring constant (zero if nodes are in proper sequence, positive otherwise) are placed between adjacent nodes in the sequence to keep them in order.

- *Cluster:* The set of nodes should be clustered together. Springs with a short rest length are placed pairwise among the nodes.

- *Zone:* The bounding box of the nodes should contain no other nodes. The bounding box is treated as a "super-node," the node-node overlap method is applied, and the resulting forces are applied equally to the nodes comprising the zone.

- *Symmetry (horizontal, vertical):* The nodes should be symmetric about the given axis. Each node is paired with the node closest to its reflection about the horizontal or vertical line through the centroid of all the nodes. (A node may be paired with itself if it is closest to its own reflection.) Equal and opposite forces are then applied to the nodes in each pair to make them symmetric about the line of reflection.

- *T-Shape:* The nodes should form a T-shape, as in a tree diagram. The user specifies which of the nodes is the parent. The T-shape VOF can be enforced as a combination of Alignment and Equal Spacing VOFs for the children and an Equal Spacing VOFs for the leftmost and rightmost children and parent.

- *Hub Shape:* The nodes should be placed radially equidistant on a circle. A central node may optionally be specified by the user; if none is specified, a phantom node (described below) is added at the center. Springs are placed between neighbors on the perimeter and between the center node and each perimeter node with rest length equal to the calculated average radius.

In addition, the following VOF can be used to gain absolute control over the fine-grained position of nodes:

- *Anchor:* A node should be located at the current position regardless of what other forces in the physical simulation may be acting on it. The Anchor VOF is implemented by calculating all forces, but then not updating the positions of anchored nodes. This is conceptually equivalent to giving anchored nodes an infinite mass.

Although the system cannot move anchored nodes, the user may still move them via the mouse. A useful technique is for a user to anchor two nodes to further constrain a second VOF. Under a Hub Shape VOF, for example, the radius of the hub is determined by calculating the average distance between the center node and each node along the periphery. To obtain a specific radius, the user could anchor the center node along with one node on the periphery. The user can then resize the hub by moving either of the anchored nodes.

Finally, GLIDE provides a single *diacritical* VOF. A diacritical VOF does not provide any constraint on the diagram, but merely augments the diagram with additional graphic elements tied to aspects of the diagram layout. The diacritical VOF implemented at present is the Frame VOF:

- *Frame:* The bounding box of the set of nodes is demarcated with a drawn frame. As the nodes participating in a Frame VOF move, the frame repositions and resizes itself accordingly. The user controls such properties of the frame as its color, padding (distance added to the bounding box before being drawn), and whether the frame is drawn as an outline or filled rectangle. The Zone VOF example in Figure 1 is illustrated using Frame VOFs.

### Constraint Satisfaction

The constraint formulation above results in a physical system consisting of a set of nodes $\{n_i\}$ and a set of generalized springs $\{s_j\}$ at any given time $t$.[6] Nodes are positioned on a canvas of typical size $1024 \times 780$, and node masses are uniformly set to 1 for ease of calculation. The spring constants $k_j$ are 1.2 or 0.0 for syntactic constraints, 0.4 for semantic constraints other than Clustering, and 0.08 for Clustering. The spring rest lengths $r_j$ are determined by the VOF conversion, as described above. The magnitude of spring $s_j$'s force is $k_j(l_j - r_j)$ by Hooke's Law, where $l_j$ is the actual length of the spring at time $t$. Node $n_i$ is therefore subject to a total force $\vec{F}_i = \sum_{j \in S(i)} \vec{f}(i,j,k_j,r_j,l_j)$ at time $t$, where $S(i)$ is the set of springs affecting node $n_i$, and $\vec{f}(i,j,k_j,r_j,l_j)$, the force vector resulting from the action of spring $s_j$ on node $n_i$, is determined by the relevant VOF conversion scheme.

Given this setup, Euler's method is used to compute the position $\vec{x}_i(t)$ and momentum $\vec{p}_i(t)$ of each node $n_i$ over time:

$t = 0$
for all nodes $n_i$
  $\vec{p}_i(t) = 0$ ;
repeat
{
  $t = t + \Delta t$ ;
  for all nodes $n_i$
    $\vec{F}_i = \sum_{j \in S(i)} \vec{f}(i,j,k_j,r_j,l_j)$ ;
  $\gamma = 2\sqrt{\max(k_j)}$ ;
  for each node $n_i$
  {
    $\vec{p}_i(t) = \vec{p}_i(t - \Delta t)$
      $+(\vec{F}_i - \gamma \vec{p}_i(t - \Delta t))\Delta t$ ;
    if $n_i$ is unanchored
      then $\vec{x}_i(t) = \vec{x}_i(t - \Delta t) + \vec{p}_i(t)\Delta t$
    else $\vec{x}_i(t) = \vec{x}_i(t - \Delta t)$ ;
  }
  if last screen update was more than $\Delta T$ ago
    then update screen to reflect positions $\vec{x}_i(t)$
}
until $\sum_i \vec{p}_i(t)^2/2 < K_{min}$ ;

The damping constant $\gamma$ for each node should ideally be the critical-damping constant for that node, but for computational simplicity, we approximate this by the critical-damping constant for the strongest spring in the system. In practice, this admittedly gross approximation has worked quite well, eliminating oscillations in almost all cases. The time increment $\Delta t$ is 0.01, and screen updates occur every $\Delta T = 0.1$ time units. When the kinetic energy goes below an arbitrary value $K_{min} = 0.01 \times |\{n_i\}|$, the simulation suspends itself temporarily until the next user action.

### Visual Presentation

By generalizing the visual appearance of graph nodes, other aspects of graph drawing can also be expressed with VOFs. For instance, *text labels* in diagrams can be implemented without additional infrastructure. A text label is merely a node with a transparent background and border. Labels can then be attached to graphical objects using a Clustering VOF for example, as in Figure 7. Similarly, a node with neither border, background, nor text is a kind of *phantom node* that can

[6] For notational simplicity we only indicate dependency on $t$ explicitly when a distinction must be made between different times, e.g., $t$ and $t - \Delta t$. Because nodes and VOFs change asynchronously in response to the user's actions and to the simulation, all the variables in the simulation are time dependent.

be useful as a control point for other objects, such as edges or Frame VOFs.[7] In particular, an orthogonal edge can be effected by interpolating a phantom node on the edge between two connected nodes and aligning the phantom node with each of the original nodes via orthogonal Alignment VOFs.

In addition to the graphical components of the diagram itself — the nodes, edges, and frames — the interface uses visual means to present to the user the current set of VOFs that are being applied to the diagram elements. Each VOF instance is indicated by a graphical *indicator* in the display; the shape of the VOF indicator is similar to the bitmap on the corresponding push button for that VOF, as shown in Figure 3.[8] As the user moves the mouse over a particular VOF indicator on the canvas, GLIDE highlights all participating nodes. Conversely, placing the mouse over a node will highlight the VOF indicators for VOF instances in which that node participates. Finally, placing the mouse over a VOF push button will highlight all VOF indicators of that particular VOF type. These mechanisms enable a user to easily determine the extent and impact of a particular set of layout constraints. The VOF graphics are intended solely to provide visual feedback to users; they are not, of course, included in the final output drawing.

GLIDE's animation is also an integral part of its visual feedback. Both user and system actions are animated as the layout is displayed in the canvas area of the system. The physical simulation continually updates the position of the nodes in the layout. Although they are not represented graphically, the forces on a node become apparent to users through observation and manipulation.

## CONCLUSIONS

Constraint-based techniques have been utilized in numerous drawing editors, e.g., [6, 8, 10, 13, 14], but they have so far enjoyed only limited success in this role. One reason for this apparent failure is the generality and complexity of most constraint-based drawing editors: the constraint vocabularies have often been chosen for orthogonality, coverage, and tractability, but may not be especially convenient or effective for particular kinds of drawing tasks. In addition, users often have difficulty understanding how a constraint-based system works, and therefore how best to utilize it. In particular, when presented with under- or over-constrained drawings, many systems behave in ways that are opaque to the user. GLIDE improves on general constraint-based systems by providing a specialized set of constraints, simple mechanisms for a user to add and delete constraints, and an intuitive method for solving the constraints. In addition, by incorporating animation into the process, GLIDE assists the user in understanding how to achieve a desired layout.

The use of a constraint-satisfaction scheme (physical simulation) that is intuitive and predictable, rather than one that is better at finding global solutions, is deliberate. GLIDE is not intended to be good at globally satisfying constraints by itself. Rather, it is intended to provide an interface that allows a useful collaboration between user and computer in solving the layout problem. For this purpose predictability, simplicity, and the compelling nature of the animation are far more important than achieving global optimality automatically. Finally, the basic concept underlying the GLIDE interface — tight collaborative interaction between user and computer to solve an optimization problem, with the computer performing local optimization and the user responsible for global control — may be applicable to other layout, drawing, and design tasks.

## ACKNOWLEDGMENTS

This work was supported by the National Science Foundation (Grant Numbers IRI-9350192 and IRI-9618848) and by MERL – A Mitsubishi Electric Research Laboratory.

## REFERENCES

1. Ronald Baecker and Ian Small. Animation at the interface. In Brenda Laurel, editor, *The Art of Human-Computer Interface Design*. Addison-Wesley Publishing Co., Reading, MA, 1990.

2. Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis Tollis. Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry: Theory and Applications*, 4:235–282, 1994.

3. Franz J. Brandenburg, editor. *Proceedings of the Symposium on Graph Drawing*, volume 1027 of *Lecture Notes on Computer Science*. Springer, 1995.

4. Ed Dengler, Mark Friedell, and Joe Marks. Constraint-driven diagram layout. In *Proceedings of the 1993 IEEE Symposium on Visual Languages*, pages 330–335, Bergen, Norway, August 1993.

5. Peter Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.

6. Michael Gleicher and Andrew Witkin. Drawing with constraints. *Visual Computer*, 11:39–51, 1994.

7. Tyson R. Henry and Scott E. Hudson. Interactive graph layout. In *Proceedings of UIST '91*, pages 55–64, Hilton Head, SC, November 1991.

[7] The use of hidden objects to control constraint-based layout has been proposed previously. See, for instance, the discussion of alignment objects by Gleicher and Witkin [6].

[8] Exaggerated forms of the VOF indicators are shown in Figures 4 through 6. In the actual system, the indicators are depicted with more visual subtlety.

8. Allan Heydon and Greg Nelson. The Juno2 constraint-based drawing editor. Technical Report 131a, Digital SRC, Palo Alto, CA, 1994.

9. Corey Kosak, Joe Marks, and Stuart M. Shieber. Automating the layout of network diagrams with specified visual organization. *IEEE Transactions on Systems, Man and Cybernetics*, 24(3):44–454, March 1993.

10. Greg Nelson. Juno, a constraint based graphics system. *Computer Graphics (Proceedings of SIGGRAPH '85)*, 19(3):235–243, July 1985.

11. Stephen North, editor. *Proceedings of the Symposium on Graph Drawing*, volume 1190 of *Lecture Notes on Computer Science*. Springer, 1996.

12. Aaron Scott. A survey of graph drawing systems. Technical Report 95-6, University of Newcastle, Australia, 1994.

13. Steve Sistare. Interaction techniques in constraint-based geometric modeling. In *Proceedings of Graphics Interface '91*, pages 85–92, Calgary, Alberta, June 1991.

14. Ivan Sutherland. *Sketchpad: a man-machine graphical communication system*. PhD thesis, MIT, 1963.

15. Roberto Tamassia and Ioannis G. Tollis, editors. *Proceedings of the Symposium on Graph Drawing*, volume 894 of *Lecture Notes on Computer Science*. Springer, 1994.

16. Bruce H. Thomas and Paul Calder. Animating direct manipulation interfaces. In *Proceedings of UIST '95*, pages 3–12, Pittsburgh, PA, November 1995.