A Physics-based Approach for Interactive Manipulation of Graph Visualizations

Andre Suslik Spritzer
Instituto de Informática
Universidade Federal do Rio Grande do Sul
Caixa Postal 15064 91.501-970 Porto Alegre, RS
Brazil
spritzer@inf.ufrgs.br

Carla M.D.S. Freitas
Instituto de Informática
Universidade Federal do Rio Grande do Sul
Caixa Postal 15064 91.501-970 Porto Alegre, RS
Brazil
carla@inf.ufrgs.br

ABSTRACT
This paper presents an interactive physics-based technique for the exploration and dynamic reorganization of graph layouts that takes into account semantic properties which the user might need to emphasize. Many techniques have been proposed that take a graph as input and produce a visualization solely based on its topology, seldom ever relying on the semantic attributes of nodes and edges. These automatic topology-based algorithms might generate aesthetically interesting layouts, but they neglect information that might be important for the user. Among these are the force-directed or energy minimization algorithms, which use physics analogies to produce satisfactory layouts. They consist of applying forces on the nodes, which move until the physical system enters a state of mechanical equilibrium. We propose an extension of this metaphor to include tools for the interactive manipulation of such layouts. These tools are comprised of magnets, which attract nodes with user-specified criteria to the regions surrounding the magnets. Magnets can be nested and also used to intuitively perform set operations such as union and intersection, becoming thus an intuitive visual tool for sorting through the datasets. To evaluate the technique we discuss how they can be used to perform common graph visualization tasks.

Categories and Subject Descriptors
H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *graphical user interfaces, interaction styles.*

General Terms
Design, Human Factors.

Keywords
Graph visualization, Interaction

1. INTRODUCTION
Graphs are present in many different application areas, ranging from biology to social network analysis and software engineering. While information organized in graph-like structures can be explored textually, through tools such as query languages, this usually requires an expert user, being too complex and not intuitive enough for most people, who have little experience with such instruments. Therefore, many of these areas make use of applications that visualize their inherent graphs in order to make it easier for their users to grasp and manipulate the information they require.

By far, the most popular and intuitive visual representation of a graph is the node-link diagram. A large community of researchers is dedicated specifically to the study of how to compute the best possible layout. The problem they attempt to solve can be simply stated as how to find the geometric positions of the nodes that are aesthetically more interesting for the better comprehension of the graph and its structure. Its solution, though, has proven to be quite complex.

Different algorithms for the layout of node-link diagrams have been created, each favoring certain aesthetic criteria, such as edge crossings, edge bends, graph symmetry, etc., in detriment of others. Some of these techniques are better for certain applications while some are better for others, but all have their limitations, which range from computational cost to visual clutter. A good source on the field of graph drawing is the book written by Di Battista et al. [2].

To deal with the limitations of these layout algorithms, many approaches have been experimented [11]. While some have applied navigation and interaction schemes on the traditional layouts, others have built 3D visualization techniques, changed from node-link diagrams to alternative visual representations, or combined existent techniques into new hybrid ones.

While some of these techniques might be well suited for certain applications, no technique is generally applicable. Also, while many techniques can build aesthetically pleasing layouts, few take into account the semantic information contained in the attributes of the nodes and edges of a graph, focusing only on the topological characteristics. Some use interaction tools such as filtering, to achieve some visual customization based on the semantic information, but very few are attribute-aware and, among those one finds that most are too application-specific to be generally applicable.

The main information associated to a graph is the relationships represented by its topology, but it is often the case that the attributes represented in its nodes and edges are just as important to the user of a graph visualization application, who might be missing out important data and even relationships that are not explicitly expressed.

AVI'08, May 28–30, 2008, Napoli, Italy.

In this work, we present a physically-based technique for the interactive manipulation of graph visualizations. Our technique consists of providing the user with virtual magnets associated with user-defined criteria, which can be topology or attribute-based, and that allow the interactive manipulation of the visualization of a graph in order to make it semantically more interesting and valuable. Magnets can also be used to intuitively perform set operations such as union and intersection, becoming a visual tool for exploring the datasets, and allowing the user to discover new relationships that were previously invisible due to the techniques that focused exclusively on the topology. To illustrate our technique, we show how it can be applied to perform common graph visualization tasks identified by Lee et al. [13].

In the following section, we present an overview of related works. Section 3 describes our approach, and gives some details of its implementation. Section 4 illustrates how it can be applied and in Section 5 we present our conclusions and draw some comments on future work.

2. RELATED WORK

The works related to the technique presented in this paper are mostly in four subjects: force-directed graph layouts; interactive graph layout reorganization; use of "false" elements for layout re-organization; and evaluation and design of graph visualization techniques.

2.1 Force-directed Layouts

Force-directed graph layouts are amongst the most popular in graph visualization due to their pleasing visual results, relative implementation simplicity and flexibility as to the inclusion of new aesthetic criteria. The basic idea of a force-directed algorithm is to treat the graph as a physical system, assigning forces to the nodes and edges, and minimizing its energy until reaching a stable layout. The forces will work to rearrange the positions of the nodes until the system finds itself in a state of mechanical equilibrium. Di Battista et al.'s book [2] presents a good survey of force-directed methods.

One of the first force-directed algorithms was proposed by Eades [6]. It takes the intuitive approach of treating the graph as a mass-spring system, with nodes being steel rings and edges springs that connect them. Despite the physical metaphor, it does not aim for physical accuracy, not employing Hooke's law for the springs and with forces affecting velocity instead of acceleration. It produces visualizations of uniform edge length and allows representation of graph symmetry. The algorithm consists of randomly positioning the nodes and simply running the simulation for a number of iterations, which is one of its drawbacks, since not all graphs converge at the same time.

Many other algorithms extended the basic idea presented by Eades. One of those is Kamada and Kawai's [12], which aims at positioning nodes in a way that their geometric distance is equal to their graph-theoretic distance. It does so by having the simulation assuming that between every two nodes there is a spring with length equal to the theoretical distance between them. Another interesting algorithm is Davidson and Harel's [5], which introduced the idea of using simulated annealing to minimize the system's energy function, which takes into account vertex distribution, edge length and edge crossings. This algorithm can be very time consuming, but can produce better results.

One of the most popular force-directed algorithms is the one proposed by Fruchterman and Reingold [8]. This algorithm consists on calculating all the forces that attract and repulse nodes, at each iteration. Nodes connected by edges exert an attraction force between them, while all nodes exert a repulsion force on all others. From the forces, the position displacement a node will suffer during each iteration is calculated and limited by the current value of an attribute (usually used as temperature), which is progressively decreased. This algorithm is relatively fast and produces nice visual results.

Another interesting approach is Noack's LinLog energy model [14], which attempts to reveal clusters of highly connected nodes. This technique is particularly useful for datasets such as social networks, and was proposed in two variations, the node-repulsion LinLog model [14][15] and the edge-repulsion LinLog model [16]. Both variations produce similar drawings, but the latter avoids dense accumulations of nodes with high degrees for graphs with non-uniform degrees.

One interesting property of force-directed algorithms is that most of them support the application of constraints. A position constraint can be established by forcing nodes to remain within a certain region, while other types of constraints can be used if they can be expressed with forces. Examples of this include the use of magnetic fields to impose orientation constraints [17] and the utilization of dummy nodes to force groupings.

For many years, force-directed algorithms have suffered dramatically from a scalability problem: the more nodes and edges we have, the slower it is for the system to converge. Thus, it was only possible to use these algorithms in real-time with smaller graphs due to their high computational cost. However, with the advent of faster, multi-core processors and powerful, programmable graphic processing units (GPUs) this reality is changing fast. Mass-spring algorithms can now deal with hundreds of thousands nodes and edges in real-time, and many different applications, such as real-time cloth simulation, are already making use of that [9, 18]. Recent works on GPU-based force-directed layout include Frishman and Tal [7], reporting a multi-level graph layout algorithm.

Aside from the scalability problem, force-directed algorithms also suffer greatly from a predictability problem. Two different runs of an algorithm over similar (or even the same) input graphs might generate two completely different layouts, which is not very helpful in allowing the user to create and maintain a mental map of the visualization. One approach that has been used to minimize this is to run another layout algorithm first, and afterwards execute the force-directed technique on that.

2.2 Interactive Layout Reorganization

Most graph visualization techniques usually use interaction and navigation techniques to explore static, pre-computed layouts. Well-known techniques include filtering; fish-eye views; scrolling and panning; zooming and even coordination of two or more visualizations (see Herman et al. [11], for a wider review on navigation and interaction techniques for graph visualizations). Very few techniques, though, allow for dynamic interactive reorganization of graph layouts.

Some applications allow for simple layout reorganization by letting the user move around nodes in force-directed layouts, which will cause an alteration in the balance of energy of the

force-directed system, thus triggering a repositioning of the nodes, which will move until equilibrium is again reached. Another known technique is to find clusters of nodes and transform them into cluster-nodes that can be expanded and collapsed by the user. Clusters can also be used to perform cluster-based semantic zooming, which allows for a level-of-detail-like approach to the visualization, letting the user incrementally explore the graph by zooming in or out.

Considering the few techniques that allow for dynamic graph layout reorganization, we find the work of Henry et al., NodeTrix [10], which is a hybrid of matrix and node-link visualizations. NodeTrix allows the user to turn clusters of nodes of node-link visualizations into matrices, which are then displayed within the node-link diagram. The layout itself is computed with the previously mentioned LinLog algorithm.

2.3 Use of False Elements

The next few related works are not exactly devoted to graph layout reorganization, but they introduced ideas that we found inspiring and somehow proved the feasibility of using magnets to allow users to re-organize visualizations in a more powerful and easy way.

Fidg't[1] is an application developed for the management of social networks that includes an interactive visualization tool that allows users to iteratively explore their networks by creating tag magnets for pictures (from Flickr) or music (from Last.fm), and observing how its nodes are attracted or repelled.

Although devoted to a different data domain (multivariate information), the Dust & Magnet information visualization technique proposed by Yi et al. [19] also uses a magnet metaphor.

Finally, it is important to mention that the technique presented in our work was partly inspired by Bier and Stone's Snap-Dragging [4], which is an interactive technique that aims at helping the user make precise line drawings.

2.4 Evaluation and Design of Graph Visualization Techniques

Evaluating such a variety of graph layout techniques and interaction techniques is a huge problem, because there are both perceptual and functional issues involved. Few works deal with evaluation of graph visualization techniques. A very useful task taxonomy for graph visualizations has been proposed by Lee et al. [13]. In their article, the authors provide a list of tasks that users might need to perform while using a graph visualization application. In Section 4 we will use this taxonomy to evaluate how our technique fares when the user tries to execute the defined tasks.

3. OUR TOOL

The goal of our technique is to aid users in interactively reorganizing the layout of a graph to better fit their needs by providing them with tools that allow the manipulation of graph visualizations based on the topological and semantic attributes that better interest them. To do so, we build on the physics metaphor of force-directed algorithms by allowing the placement of virtual magnets, which attract nodes that fulfil certain user-defined criteria. While we follow a magnet metaphor, though, physical accuracy is not one of our aims.

In our technique magnets can be placed on the scene in order to reorganize the graph. They can be set to attract nodes based on their values for certain criteria, which can be topological (such as nodes that have a certain degree or that have a path to another node with a certain length) or attribute-based (i.e. all users that come from the UK). Also, boundary shapes can be applied to magnets to keep the nodes they attract bound to certain regions of the scene.

3.1 Basic Graph Layout

In our technique, the user is given two options for the computation of the layout: a mass-spring algorithm similar to Eades's [6] or an adapted version of Fruchterman and Reingold's technique [8].

In the first option, the layout of the graph is computed assuming that all nodes have equal mass. The parameters of the algorithm, such as time step, edge rest length, damping factor and node repulsion force can be changed by the user to produce a visualization that is more satisfying aesthetically. The rest length of the edges can be either a fixed value provided by the user or computed based on the degree of the nodes that it links (the higher the degree, the longest the length). The layout is dynamic and is always being recomputed; therefore, any change in the position of a node will trigger a subsequent reorganization of the layout. Figure 1 shows a small graph with layout computed using this algorithm.

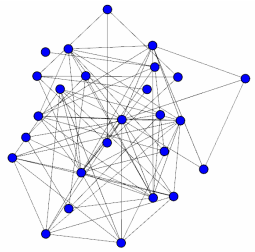

Figure 1. Graph of the largest component (largest connected subgraph) of the AVI coauthorship network drawn using a mass-spring algorithm.

In the second option, Fruchterman and Reingold's technique is combined with the Barnes and Hut algorithm [3], and slightly adapted to better fit our needs. Our modifications were the addition of a small gravitational force that pulls all nodes slightly towards the centre of the workspace and the alteration of the manner in which the algorithm runs (we re-evaluate it every frame instead of running it for a given number of iterations). The user can set several parameters, such as time step, damping, a constant that is used for the computation of the optimal distance between two vertices, maximum attraction force (to make it easier for the simulation to reach stability), attraction and repulsion exponents and central gravitation factor. Figure 2 shows the same graph as Figure 1 computed using this algorithm.

[1] http://fidgt.com

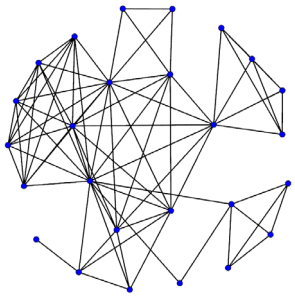

Figure 2. Graph of the largest component of the AVI coauthorship network drawn using our variation of Fruchterman and Reingold's technique.

In both cases, Verlet integration is used to compute node positions in every frame due to its stability and area preserving properties. To allow for easier navigation, the user can pause and resume the simulation at any time.

3.2 Magnets

Magnets are special objects that can be added to the scene which have the ability to attract nodes of a graph that fulfil certain user-defined criteria. Figure 3 shows how magnets are visually represented in the prototype we developed.

Figure 3. Visual representation of a magnet.

A magnet works by exerting onto each of these nodes an attraction force that will progressively move them towards it, thereby building a cluster of semantically-related nodes around it. When these nodes move, the force-directed layout algorithm ensures that all the other nodes that are connected to them by edges will be pulled along, reorganizing the whole layout of the graph in the process.

To each magnet users should associate one or more attraction criteria, which can be set as requirements of attraction or simply criteria. To be attracted a node must fulfil all requirements and at least one of the defined criteria. These requirements and criteria can be based on the topology of the graph, the attributes of its nodes and edges or even other magnets that have been placed on the scene.

Topology-based criteria use the structure of the graph to attract nodes. It is possible to attract nodes based on properties such as degree, path length (i.e. all nodes that are within a specified path length from another node or group of nodes), connected subgraph (i.e. subgraphs with a given number of nodes), connected components (maximally connected subgraphs with a given number of nodes) . Figure 4 shows an example of two magnets with topology-based criteria in action.

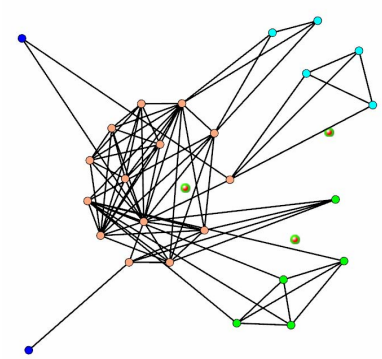

Figure 4. A small graph with three magnets - one targeting nodes of degree 3 (light blue nodes); the second attracting nodes of degree 4 (green nodes) and the third one attracting nodes with degree greater than 5 (light orange nodes).

Attribute-based criteria use the semantic properties contained in nodes and edges in order to attract nodes. Users can set a magnet to attract all the nodes in which a certain property exists, or not only exists but is also equal to a certain value or is within a certain value range (if it is numerical). Users can do the same for edges, with the magnet then attracting the nodes linked by edges that fulfil the defined criteria. An example of a magnet with an attribute-based criterion can be observed in Figure 5.

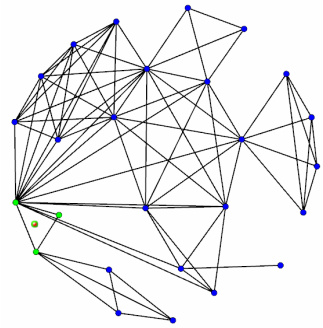

Figure 5. Graph of the largest component (largest connected subgraph) of the AVI coauthorship network with a magnet set to attract authors cited more than once (all nodes that have attribute "citationsnb" greater than 1).

Magnet-based criteria use the sets of attracted nodes of each magnet that was included in the scene by the user. With magnet-based criteria, one can set a magnet to attract all the nodes that another magnet also attracts, all the nodes that another magnet does not attract, all the nodes that no magnet attracts or all the nodes that are attracted by a combination of magnets. This allows for set-based operations on the graph visualization, which usually will end up in a graph reorganization.

Each criterion has a properties dialog through which it can be properly set up and configured. They can be added, removed and edited at any time, with users also being able to add multiple criteria and requirements to each magnet. This makes it possible, for instance, to set a magnet to attract all nodes that are not attracted by any magnet, have degree higher than a certain number and include the property that is called *x*. Figure 6 shows an example of a magnet with different types of criteria.

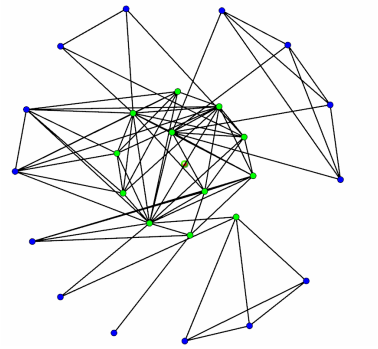

Figure 6. Magnet attracting authors with at least two papers that have written papers with more than three other authors (nodes with the attribute "papersnb" greater than or equal to 2 and degree greater than 3).

In case one might wish to perform a union of the set of nodes attracted by two or more different magnets, it is possible to combine such magnets. The combination operation will create a new compound magnet with the combined criteria of the ones selected. The new magnet will have its force magnitude set to the average of the original ones', which will be subsequently set to 0.

Within the physics metaphor, a magnet works simply, on each frame, by applying to all attracted nodes a force vector in its direction with the specified magnitude. Also, to keep the magnet from being overlapped by its attracted nodes and to keep the attracted nodes from staying all bundled together too close to each other, the magnet also exerts a repulsion force on each of the attracted nodes. This repulsion force is the same as with common nodes, working like a reverse gravity by being inversely proportional to the distance of the node to the magnet and proportional to the magnitude of the force of attraction, so that it is stronger with the nodes that are near the magnet and weaker with the ones that are progressively further away from it. The magnitude of the repulsion force can be increased by the user to change the minimum distance the nodes ought to have from the magnet.

Occasionally it might be cumbersome to see which nodes that are positioned close to a magnet are in fact attracted by it. To deal with this situation, it is possible to assign a colour to all the nodes that it attracts or to create a boundary shape around the magnet to limit the region in which such nodes can move about.

3.3 Boundary Shapes

A boundary shape is simply a geometric shape (a circle in our current implementation), which can be placed around a magnet and have the function of bounding the nodes that such magnet attracts to the region that the shape delimits. At the same time that all attracted nodes are kept within the boundary shape, all other nodes are kept out, with the shape exerting a repulsion force similar to the one exerted on the other nodes by the nodes themselves (a reverse gravity force).

To allow for a better distribution of space within a boundary shape, the magnitude of the attraction force of the magnet is reduced for the nodes that are inside. Also, when a node enters the region of a boundary shape, the direction of the force that pulls it is "refracted" by the application of Snell's Law. So, instead of there being a force that pulls the nodes straight to the magnet, each node is pulled towards a nearby direction, which makes for more evenly distributed nodes. Figure 7 shows boundary shapes in action.

Once a node finds itself inside a magnet's boundary shape, it cannot escape that area, unless it is also attracted by another magnet that is placed outside such shape.

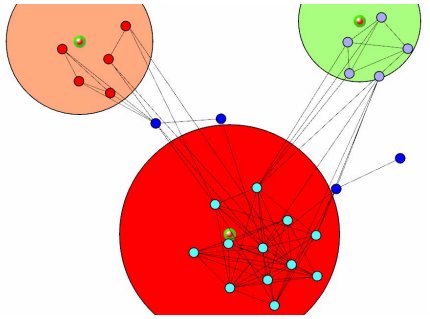

Figure 7. Graph with boundary shapes.

3.4 Magnet Hierarchy

A magnet effectively creates sets of related nodes and ensures that they remain near a certain physical region. Occasionally it might be useful to refine this set of nodes into subsets. To allow for that, it is possible to define magnets that act only on the subset of the graph that is already attracted by another magnet. To do this, the user must simply create a magnet and define another one as its parent. It is interesting to note that children magnets might children magnets of their own; creating thus a hierarchy of magnets that might be helpful for incremental exploration of a graph. Figure 8 illustrates the use of magnet hierarchy to achieve a better organization of the layout.

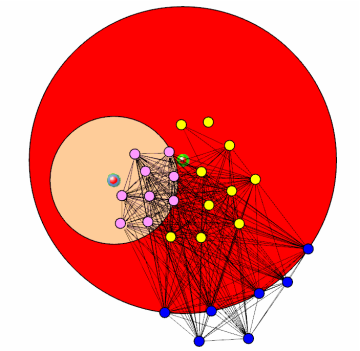

Figure 8. Graph of the CHI conference citations with a magnet hierarchy. All nodes within the red boundary shape are proceedings of the conferences that were published before 2000, with the ones inside the beige shape being from after 1990.

If the parent magnet does not have a boundary shape, or has one, but the child magnet is outside of it, a dashed line in the same colour as the parent's nodes appears between them. If there is a boundary shape and the child magnet is within it, no line appears.

3.5 Magnet Intersections

Figure 9. Visual representation of an intersecting node.

Occasionally it may happen that a node fulfils the criteria of two or more magnets. In such a case, the node will be attracted to all of these magnets and will thus have a tendency to stabilize in the middle of them with a bigger lean towards the ones with the strongest attraction forces. If there is intersection between magnets that have boundary shapes, their position constraints are ignored, and they are allowed to escape the region they were previously bound to.

To make the intersections more apparent visually, the common nodes are drawn as in Figure 9. The user may also at any time choose to display dashed lines from the intersecting nodes to their 'parent' magnets (Figure 10). Each line assumes the colour that was defined by the user for the nodes that are attracted by its respective magnet.

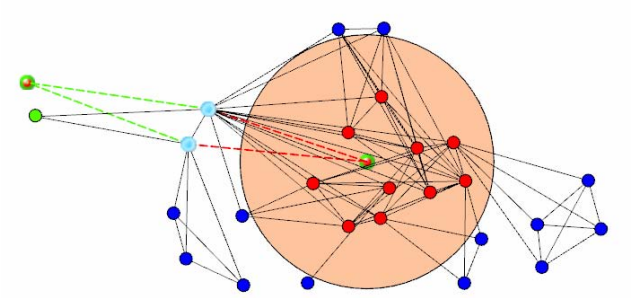

Figure 11. Largest component of the AVI coauthorship network with a magnet that attracts all authors that have more than one paper (node attribute "papersnb" greater than 1) and another that attracts all authors with more than one citation (node attribute "citationsnb" greater than 1).

3.6 Implementation Details

A proof-of-concept prototype was developed to test and evaluate our approach. It was initially developed with Python 2.5 and Qt 4.3.1, using PyQt and later ported to C++. The prototype takes as input GraphML files and displays the graphs with the previously described layouts.

Users are able to insert magnets, whose attributes (including shapes and criteria) can be manipulated through a panel on the right side of the graphical user interface. Each criterion has its own properties dialog, which can be accessed by picking it from the selected magnet's requirement and criteria lists. The prototype was built with extensibility in mind, so that the creation of new tools and types of criteria is straightforward.

The panel on the right side of the user interface is used to provide layout and magnet options to the user. When no node or magnet is selected, information about the graph and the layout configuration interface is displayed. If a magnet is selected, the magnet editor is launched, allowing the user to set and edit the magnet criteria and boundary shape. When the user selects a node, the panel displays the attributes of that node.

As the goal of this prototype was to test our technique, performance considerations were not taken into account in the development of the application. Therefore, the current implementation is completely on software and with only few optimizations. Nevertheless, on the machine used to run it, a single-core Mobile AMD Athlon 64 3000+ (2.0 GHz) with 2 GB of DDR 333 MHz memory and an ATI Radeon 9700 graphics card with 128 MB of memory, it was already possible to deal with graphs of several hundred nodes and edges at interactive rates.

4. DISCUSSION

Lee et al. [13] have proposed a useful task taxonomy for graph visualization in which it is defined a list of tasks that are commonly performed while exploring a graph. They divide these tasks into general low-level tasks, graph-specific tasks and complex tasks, with the latter being further categorized into topology, attribute-based, browsing and overview tasks.

To examine how our technique can contribute to the visualization of a graph, in this section we show how it can be used to better carry out many of the tasks on Lee et al.'s taxonomy.

Table 1. Low-level tasks inherently covered by our technique

Task	Description
1. Filter	Given some conditions on attribute values, find data cases satisfying those conditions.
2. Find Extremum	Find data cases possessing an extreme value of an attribute over its range within the data set.
3. Sort	Given a set of data cases, rank them according to some ordinal metric.
4. Determine Range	Given a set of data cases, rank them according to some ordinal metric.
5. Characterize Distribution	Given a set of data cases and a quantitative attribute of interest, characterize the distribution of that attribute's values over the set.
6. Find Anomalies	Identify any anomalies within a given set of data cases with respect to a given relationship or expectation.
7. Cluster	Given a set of data cases, find clusters of similar attribute values.
8. Correlate	Given a set of data cases and two attributes, determine useful relationships between the values of those attributes.
9. Find Adjacent Nodes	Given a node, find its adjacent nodes.
10. Set Operation	Given multiple sets of nodes, perform set operations on them.

Most of the higher-level tasks are built on combinations of the 10 general low-level visual analytic tasks described by Amar et al. [1] and also three other operations proposed by themselves (with one of them being exclusive to graphs). It is interesting to note how our technique already inherently deals with several of these lower-level tasks. Table 1, partially taken from Lee et al.'s paper,

contains a listing and description of the low-level tasks that our approach is able to cover.

As can be clearly seen from Table 1, these tasks can be performed through our technique simply by adding magnets to the scene with the proper combination of criteria, and allowing the graph to reorganize itself. Tools such as magnet boundary shapes and the ability to operate on magnets themselves (through magnet combination and magnets that have magnet-based criteria) make carrying out these tasks a natural fit, with clustering and set operations being some of the most natural applications for the tools we propose.

Regarding to graph-specific higher level tasks, our technique also provides adequate support to the user in accomplishing several of them. In most cases the tasks can be easily carried out by relying simply on the placement of magnets with the proper combination of topology and attribute-based criteria followed by (if necessary) the proper operations on the magnets themselves (such as magnet-based criteria and magnet combination).

Lee et al. divide graph complex tasks into topology-based tasks, attribute-based tasks, browsing tasks and overview tasks. Our technique is especially useful for the first two categories and can be easily integrated into applications that provide ways to accomplish the other two types of tasks.

Topology-based tasks were further subdivided into a few categories: adjacency, accessibility, common connection and connectivity. Adjacency tasks include finding the set of nodes adjacent to a node, a node's degree and the node with the highest degree. Accessibility includes issues such as finding all the nodes accessible from another one and the set of nodes with distance from another node within a certain range. Common connection corresponds to finding a set of nodes that are connected to all the nodes of a given set, while connectivity includes finding the shortest path between two nodes, finding connected components (defined by Lee et al. as a maximal connected subgraph) and clusters (defined by the same authors as a subgraph of connected components whose nodes have high connectivity).

Attribute-based tasks can work on nodes or edges and include operations such as finding the nodes that have a specific attribute value or that are linked by edges that have a certain attribute or a certain attribute value in a specified range.

Browsing tasks include operations such as following a given path or revisiting a previously visited node.

Finally, overview tasks correspond to exploratory operations performed in order to quickly get an estimate of a certain value, such as the size of a graph or subgraph, or patterns that the graph tends to have.

As can be seen from the previous description of the different types of tasks, magnets apply directly to topology and attribute-based tasks. Such tasks can be accomplished simply by creating magnets with the proper criteria. Browsing and overview tasks can also be helped by the magnets, by making it easier to find nodes on the scene and providing the visualization with some node position predictability, since magnets can be inserted to make sure that nodes that fulfill certain criteria are within a certain region. For the tasks that our tools are unable to cover, the solution is simply a matter of combining it with other techniques, such as fish-eye-like visualizations, overview windows, node search, etc.

One interesting aspect of our technique is that it can be used to easily explore graph datasets by building queries through the specification of magnets and their criteria, and performing set operations on them, becoming thus an intuitive and simplified alternative to query languages or filtering operations, which can be too complex for most end-users that do not have advanced programming and computer skills.

5. CONCLUSIONS AND FUTURE WORK

Even though there is a multitude of graph layout algorithms, there is no one which fits to all types and sizes of graphs. With the work presented in this paper, we aimed at developing a technique that would help circumvent this fact by providing the user with tools that could allow him to shape a layout into one of his/her needs.

On the contrary of most graph layout techniques, which work solely based on the topological structure of the graph, ours also takes into account the information contained in attributes of the nodes and edges. This allows the user to dispose the graph in a layout that can be semantically more interesting.

Our tools, in great part due to the metaphor we employ, make it possible for the user to intuitively navigate through the graph and perform many common graph visualization operations.

One of the biggest drawbacks of force-directed algorithms is that the layouts they produce tend to be unpredictable – different runs on similar graphs (or even with the same one) might generate completely different layouts, which is quite a hindrance for maintaining a mental map of the graph. Our technique helps minimize this limitation, allowing for a level of predictability in otherwise unpredictable drawings. In two runs of the application on the same graph, two magnets will always attract the same nodes to the same place. It is not guaranteed that the nodes will be at the same exact position, but their general location can be easily known, since it is indicated by the users themselves.

Another interesting aspect of the presented technique is that it is not bound to a specific layout algorithm: it can work with any that allows for forces to be applied to nodes.

There is still work to be done in order to improve the technique presented in this paper. Amongst the planned work is an efficient implementation of the technique using the GPU on top of a more sophisticated layout algorithm that more clearly separates clusters of highly connected nodes, such as LinLog [14]. This implementation will allow the use of the technique with larger and more complex datasets, permitting its validation and better adaptation for huge graphs, which have shown up quite frequently lately due to the growing interest in the visualization of social networks.

Some new features are also planned for the technique itself, such as new types of criteria, arbitrarily-shaped magnet boundaries, the possibility of making a magnet work only on the nodes that are within a certain area (i.e. with a certain radius around it), and the ability to collapse the nodes attracted by a magnet into an expandable and collapsible cluster-node to allow for a better iterative visualization. Also planned is a special magnet that applies weighed forces to the nodes it attracts, allowing for a visual sorting of such nodes (the closer the node is to the magnet, the more it has of a certain property). This sorting magnet would

allow operations such as visually browsing by date, alphabetically or any numerical value.

Planned work also includes user experiments for better validation of the technique as well as its integration into a complete graph visualization application that supports other features such as node search, overview windows, coordination with different visualizations, filtering and fish-eye-like views.

6. ACKNOWLEDGMENTS

We gratefully acknowledge the helpful comments of our colleagues as well as Jean-Daniel Fekete and Nathalie Henry. We also thank J.-D. Fekete and N. Henry for providing the datasets we use in the paper. This work is partially sponsored by CNPq (Brazilian Council for Research and Development), specially the CNPQ/INRIA cooperation program (grant nr. 490087/2005-1).

7. REFERENCES

[1] Amar, R., Eagan, J. and Stasko, J. 2005 Low-Level Components of Analytic Activity in Information Visualization. In Proceedings of 11th IEEE Symposium on Information Visualization (2005), 111-147

[2] Battista, G., Eades, P., Tamassia, R., and Tollis I.G. 1999. Graph Drawing: Algorithms for the Visualization of Graphs. Prentice Hall, New Jersey.

[3] Barnes, J. and Hut, P. 1986. A hierarchical O(N log N) Force-calculation Algorithm. Nature, 324(4).

[4] Bier, E. and Stone, M. 1986 Snap-dragging. ACM Computer Graphics, 20 (August 1986), 233-240.

[5] Davidson, R. and Harel, D. 1996. Drawing Graphs Nicely Using Simulated Annealing, ACM Transaction on Graphics, 15 (4), 301–331.

[6] Eades, P. 1984. A Heuristic for Graph Drawing, Congressus Numerantium, 42(1984), 149–160.

[7] Frishman, Y. and Tal, A. 2007. Multilevel Graph Layout on the GPU. IEEE Transactions on Visualization and Computer Graphics, 13 (Nov-Dec 2007), 1310-1319.

[8] Fruchterman, T.M.J. and Reingold, E.M. 1991. Graph Drawing by Force–Directed Placement. Software - Practice & Experience, 21 (Nov 1991), 1129–1164.

[9] Georgii, J., Echtler, F. and Westermann, R. 2005 Interactive simulation of deformable bodies on GPUs. In Proceedings of Simulation and Visualization, 2005, 247- 258.

[10] Henry, N., Fekete, J.-D. and McGuffin, M. 2007 NodeTrix: Hybrid representation for analyzing social networks. IEEE Transactions on Visualization and Computer Graphics, 13 (Nov-Dec 2007), 1302-1309.

[11] Herman, I., Melancon, G., and Marshall, M. S. 2000.Graph visualization and navigation in information visualization: A survey. IEEE Transactions on Visualization and Computer Graphics, 6 (Jan-Feb 2000), 24-43.

[12] Kamada, T. and Kawai, S. 1989. An Algorithm for Drawing General Undirected Graphs, Information Processing Letters, 31(1989), 7–15.

[13] Lee, B., Plaisant, C., Parr, C., Fekete, J-D., and Henry, N. 2006. Task taxonomy for graph visualization. In Proceedings of the 2006 AVI workshop on BEyond time and errors. (Venice, Italy) BELIV 2006, ACM Press, New York, NY, 81-86, DOI= http://doi.acm.org/10.1145/1168149.1168168.

[14] Noack, A. 2003. Energy Models for Drawing Clustered Small-World Graphs. Technical Report 07/03, Computer Science Reports, Brandenburg University of Technology at Cottbus.

[15] Noack. A. 2004. An Energy Model for Visual Graph Clustering. In Proceedings of the 11th International Symposium on Graph Drawing (Perugia, Italy, Sep. 21-24), GD 2003, Springer-Verlag, Berlin, LNCS 2912, 425-436.

[16] Noack. A. 2005 Energy-Based Clustering of Graphs with Nonuniform Degrees. In Proceedings of the 13th International Symposium on Graph Drawing (Limerick, Ireland, Sep. 12-14), GD 2005, Springer-Verlag, Berlin, LNCS 3843, 309-320.

[17] Sugiyama, K. and Misue, K. 1995 A Simple and Unified Method for Drawing Graphs: Magnetic-Spring Algorithm. In Proceedings of the International Workshop on Graph Drawing, (Princeton, NJ, USA, October 1994), GD'94, Springer-Verlag, Berlin, LNCS 894, 364-375.

[18] Tejada, E. and Ertl, T. 2005 Large Steps in GPU-based Deformable Bodies Simulation. Simulation Modeling Practice and Theory, 13(2005), 703-715.

[19] Yi, J.S., Melton, R. Stasko, J., and Jacko, J. 2005 Dust & Magnet: multivariate information visualization using a magnet metaphor. Information Visualization, 4 (2005), 239-256.