International Journal of Modern Physics C
Vol. 18, No. 10 (2007) 1537–1549

World Scientific
www.worldscientific.com

SELF-ORGANIZATION APPLIED TO DYNAMIC NETWORK LAYOUT

MARKUS M. GEIPEL
Chair of Systems Design, ETH Zurich
8032 Zurich, Switzerland
mgeipel@ethz.ch

Received 9 November 2006
Accepted 18 April 2007

As networks and their structure have become a major field of research, a strong demand for network visualization has emerged. We address this challenge by formalizing the well established spring layout in terms of dynamic equations. We thus open up the design space for new algorithms. Drawing from the knowledge of systems design, we derive a layout algorithm that remedies several drawbacks of the original spring layout. This new algorithm relies on the balancing of two antagonistic forces. We thus call it *arf* for "attractive and repulsive forces". It is, as we claim, particularly suited for a dynamic layout of smaller networks ($n < 10^3$). We back this claim with several application examples from ongoing complex systems research.

Keywords: Networks; self-organization.

PACS No.: 89.75.Fb.

1. Introduction

In many disciplines — sociology, physics, mathematics, economics, biology — networks and their structure have become a cross-cutting concern.[1,8,17] So far, research focused mostly on statistical properties of these networks. A new challenge is the analysis of dynamic aspects of networks. In such networks, links are created and vanish, nodes are added and dropped. For both the analysis of the structure and that of the dynamics of networks, visualization is an invaluable tool.

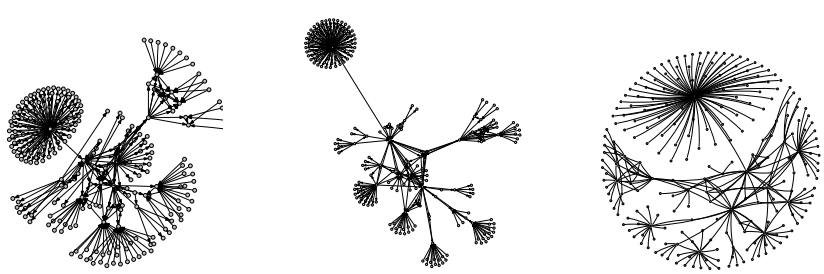
A very appealing and successful approach to the network layout problem is the so-called *force directed layout*. It was first proposed in 1984[4] and is related to the concept of self-organization based on only local interactions. In fact, many natural structures become functional, efficient and even visually appealing by self-organization. Often balancing two antagonistic forces gives rise to emergent order. Balancing, for example, activation and inhibition in an artificial chemistry was used to mimic the formation process of patterns on sea shells.[12] Another example is the modeling of vortex swarming of Daphnia:[11] attractions ensure the coherence of the

swarm while repulsion prevents it from collapsing. Finally, a force model can be employed to position soccer playing robots on the field: The robots feel attracted to the ball and the goal and feel repelled by other robots.[18] In the same vein, a network can be regarded as a system comprised of agents represented by the nodes and their interactions represented by links between them. In *force directed layout*, a combination of simple forces leads to the emergence of a global spatial structure.

The most commonly used class of such algorithms is the spring layout. It was, for example, used to dynamically explore the structure of the WWW.[2] In the spring model, nodes experience a spring force (Hooke's law), that adjusts the distances of connected nodes and a repulsion (Coulomb's law) to spread out unconnected nodes. The mechanical equilibrium is supposed to possess favorable properties such as low edge crossings and nearly equal edge length. It can be searched in two different ways. The first option is to simply simulate the system. This is also the only way to generate a dynamic network layout. The second option is to search for a local energy minimum more directly by general global optimization methods. As we deal with dynamically changing networks, we will only consider the first method.

The most prominent representatives of spring layout are the Kamada–Kawai[7] and the Fruchterman–Reingold[5] algorithm. Figures 1(a) and 1(b) show a complex heterogeneous network visualized with them. Several problems spring to mind: congestion, orderless arrangement of nodes in the highly connected core and high variation in the edge length.

The contribution of this paper is first, to analyze the shortcomings of the spring model and second, to present a modified model that addresses these shortcomings. We will proceed as follows: First, we will formalize and analyze the spring model in its most general from. Having opened up the design space we propose changes to remedy the spring model's defects. So, a new model will be derived from the general spring model. In the next section, an evaluation of the new model will be presented. This will be followed by several application examples in the field of complex systems. Finally concluding remarks will round up the discussion.



(a) (b) (c)

Fig. 1. Network showing the relationships of managers based on their board memberships. Layout of the same network with *neato* on the left (a), ftp in the middle (b) and layout by *arf* on the right (c). For clarity, labels were omitted.

2. The Spring Model: A Critical View

For the following discussion we will assume a graph G to be a tuple $G = (V, E)$ where V is the set of vertices or nodes respectively and $E \subseteq V \times V$ is the set of edges. We will use the indices i and j to denote vertices. Whether the graph is directed or undirected is irrelevant for the discussion.

The general spring model is based on the idea that springs adjust the distances between connected nodes: connected nodes should be placed equidistantly. The spring pushes towards this desired distance d which is the resting position of the spring. The force on i of a spring between i and j is given as:

$$m\ddot{x}_i = -K_{i,j}(|x_i - x_j| - d)\frac{(x_i - x_j)}{|x_i - x_j|} \tag{1}$$

x_i is used to denote the position vector of node i. $K_{i,j}$ is the spring constant of the spring connecting i and j. Thus K is defined by:

$$K_{i,j} = \begin{cases} k\,, & \text{if } (i,j) \in E \vee (j,i) \in E\,; \\ 0\,, & \text{otherwise}\,. \end{cases} \tag{2}$$

As only connected nodes have springs between them, each node is endowed with a repulsive field to adjust their distances. The resulting force experienced by i in the presence of j is thus:

$$m\ddot{x}_i = \rho\frac{(x_i - x_j)}{|x_i - x_j|^{1+\beta}}\,. \tag{3}$$

Assuming that we mimic a Coulomb field, ρ can be interpreted as the electrostatic constant multiplied by the charges of the two nodes. β is commonly set to 2.

Unfortunately, a system such as this will start oscillating. Therefore friction is introduced:

$$m\ddot{x}_i = -\gamma\dot{x} \tag{4}$$

γ denotes the friction constant.

Putting these forces all together, we get the following equation:

$$m\ddot{x}_i = -\gamma\dot{x} - \sum_{j\in V} K_{i,j}(|x_i - x_j| - d)\frac{(x_i - x_j)}{|x_i - x_j|} + \rho\sum_{j\in V}\frac{(x_i - x_j)}{|x_i - x_j|^{1+\beta}} \tag{5}$$

under the assumption that $m \ll \gamma$, ρ, K and $\gamma = 1$ we can derive the following approximation for the movement of the nodes.

$$\dot{x} = -\sum_{j\in V} K_{i,j}(|x_i - x_j| - d)\frac{(x_i - x_j)}{|x_i - x_j|} + \rho\sum_{j\in V}\frac{(x_i - x_j)}{|x_i - x_j|^{1+\beta}}\,. \tag{6}$$

Having presented the general formulas for spring layout we will take a look at the outcome. Figure 1(a) shows a network, visualized with the spring layout by Kamada and Kawai[7] as implemented in graphvis' *neato* layouter. Figure 1(b) shows the same network, visualized with an alternative spring model proposed by

Fruchterman and Reingold[5] (graphvis' *fdp* layouter). We notice several problems: First, there is congestion around strongly connected nodes (center area and bottom right). Too many nodes are crammed in too little space by a force too strong. Unfortunately, the strongly connected nodes are often the ones that are of special interest in network analysis. Second, the available layout space is used inefficiently: there are big empty spaces, while in the center some nodes are occluded by others in the case of Fig. 1(a). An even distribution of nodes moreover would facilitate any labeling of nodes. For Fig. 1(b) the structure is clearer. However the star cluster is pushed too far off by its repulsive field. The one connecting spring is not sufficient to compensate the repulsion. This unbalance is especially distinct in networks with highly heterogeneously node degrees such as the ones predominant in complex systems research. Finally, the layout does not show much symmetry or structure. Instead, especially in the center, it leaves a rather chaotic impression. The self-organizing forces do not seem to be strong to disentangle the network. However, for laying out dynamically changing graphs, this would be vital.

The basic problem is this: The different forces at work are not well tuned. In some situations, attraction is too strong, in others repulsion is too dominant.

3. A New Model

The new model centers around the principle of balancing two forces: An attractive and a repulsive one. We thus call it *arf*. We search for forces that better represent commonly accepted principles of graph design such as the following ones:[5]

(1) Vertices connected by an edge should be drawn near each other.
(2) Vertices should not be drawn *too* close to each other.

The fact that "[$\cdots$] the layout should display as much symmetry as possible" (as noted by Ref. 4) is also taken into account. In doing this, we will derive a new force model for laying out networks.

Let us first take a look at the spring force: Obviously, connected nodes should be close to each other for two reasons: First humans find it difficult to follow long edges. Thus, short ones make the graph more readable. Second the shorter an edge, the lower the risk of crossing another one. Thus an attractive spring force between nodes makes perfect sense. But, all nodes of a graph should stay close together to ensure that they are evenly spread. This will avoid runaway clusters such as the one in Fig. 1(b) because each repulsive connection between two nodes is now *balanced* by an attractive one. The new equation for the spring constant K is defined as follows:

$$K_{i,j} = \begin{cases} 0\,, & \text{if } i = j\,; \\ a\,, & \text{if } (i,j) \in E \vee (j,i) \in E\,; \\ 1\,, & \text{otherwise}\,. \end{cases} \tag{7}$$

The parameter a gives the strength of springs between connected nodes. It has to be greater than 1: $a > 1$. The greater a, the clearer the separation of unconnected sub clusters.

As described in the previous section, the resting position d reflects the desired edge length. The spring force can be both attractive and repulsive, depending on the position of the nodes. However, there is already a repulsive field. This means that we have two possible repulsive forces. By setting $d = 0$ in Eq. (1) we accomplish two things: First, we clearly separate the attractive and repulsive forces. A more straightforward force system is easier to tune. Second, we abandon the rule to enforce equal length edges which led to congestion around highly connected nodes. The new equation for the attractive force is simply

$$m\ddot{x} = \sum_{j \in (V \setminus i)} K_{i,j}(x_i - x_j)\,. \tag{8}$$

Besides the concern to have connected nodes close together and edges to be short, an equal distribution of nodes in the available layout space is also desirable. Repulsive movement addresses this concern. The balance between attraction and repulsion regulates the edge length in a very flexible way. However, in the previous section we diagnosed an imbalance in the force system. Loosely connected clusters were driven away by too much repulsion and in highly connected clusters, repulsion was not strong enough to counter congestion. With the new function for K we already addressed this issue. Yet, we can still improve by changing repulsion from a quadratically decaying force to a distance invariant one. This is done by choosing $\beta = 0$. The following term is used to calculate repulsion:

$$m\ddot{x}_i = -\rho \sum_{j \in (V \setminus i)} \frac{x_i - x_j}{|x_i - x_j|}\,. \tag{9}$$

Next, we have to take into account changing graph sizes. The node density should be invariant, not the size of the layout space. To scale the layout space based on the number of nodes in the graph, the repulsive force is multiplied by $\sqrt{|V|}$. This is accomplished by defining

$$\rho = b\sqrt{|V|}\,, \tag{10}$$

where b in Eq. (10) scales the radius of the layout space.

When combining the force Eqs. (8)–(10), and the friction we can derive the following movement approximation for the nodes:

$$\dot{x}_i = \sum_{j \in (V \setminus i)} \left(K_{i,j} - \frac{b\sqrt{|V|}}{|x_i - x_j|} \right) (x_i - x_j)\,. \tag{11}$$

The equation always leads to a circular layout space for the graph which is also very convenient for generating *animations* as frame sizes are invariant. Node density can be adjusted with the parameter b and the separation of unconnected nodes with the parameter a in the equation for K. The complete layout procedure is given by

algorithm (1). The update of the node-position takes place in line 5 of algorithm (1). The parameter Δt adjusts the size of one time step and thus the precision of the numerical integration. There is a tradeoff between speed and potential instability resulting from too large values of Δt. The update is repeated until the amount of changes in the graph drops beneath a certain threshold ϵ.

Algorithm (1)

1: **Repeat**
2: $error \leftarrow 0$
3: **for all** $i \in V$ **do**
4: $\dot{x}_i = \sum_{j \in (V \setminus i)} (K_{i,j} - \frac{b\sqrt{|V|}}{|x_i - x_j|})(x_i - x_j)$
5: $x_i \leftarrow x_i + \Delta t \dot{x}_i$
6: $error \leftarrow error + |\dot{x}_i|$
7: **end for**
8: **until** $error < \epsilon$

---

4. Evaluation

In this section we provide an evaluation of the new layout rules. As there is no commonly accepted benchmark for network layout, the following evaluation cannot and does not attempt to provide hard evidence. Rather it attempts to indicate advantages of the movement model in a clear cut situation: layout of small to medium size networks.

As a reference point for comparisons, the layouters contained in the *graphviz* package[a] were used. *Graphviz* is maintained by AT&T and is probably the best currently available graph layout tool. It contains the following five layouters: *dot*,[6] *neato*,[7] *circo*,[9,16] *twopi*[19] and *fdp*.[5] The following evaluation compares *arf* with respect to layout quality. Computational complexity is not the issue here. Suffice it to say that all force directed layouts reside in the same complexity class, which is assumed to be $O(|V|^2)$. There are techniques to achieve lower complexity.[b]

When assessing layout quality, the following four criteria are commonly accepted:

(1) minimal number of crossing edges
(2) aesthetics of the layout

[a]Available under the Common Public License at `http://www.graphviz.org/`.
[b]For more detailed discussions see Refs. 3–5, 7 and 14.

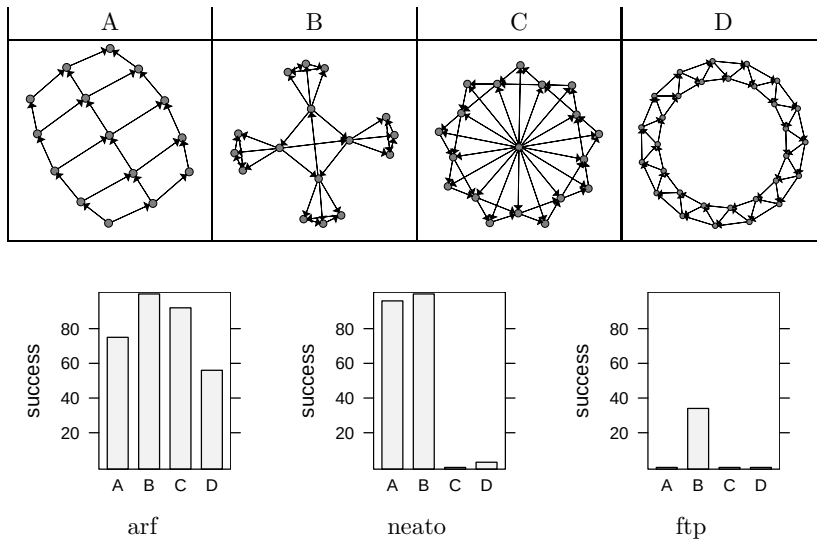


Fig. 2. Top: the four graphs (A, B, C and D) were used as benchmark. Bottom: Percent of successful layouts of each layouter for each of the benchmark graphs. For details see the text.

(3) separation of clusters
(4) usage of the available space

First, we will address points one and two: edge crossings and aesthetics. As the initial node positions are random, just looking at *one* generated layout is not enough. Thus, the following benchmark was used: The task is to lay out the four symmetrical graphs shown in Fig. 2 for 100 times. The layout in the figure was produced by *arf*. Besides *arf*, the evaluation takes into account the layouters *neato* and *fdp*, both of which are also probabilistic and force directed. The criteria for success are quite simple: For graphs A and D, each symmetric layout without crossings is counted as a success. For graphs B and C, no algorithm finds a crossing-free layout. So, all symmetric layouts that have no more crossings than the best produced layout will be accepted. Figure 2 shows the results. It can be seen that *arf* applied to a random node distribution, on average, finds the optimal layout more often than *neato* and *fdp*, even though they use more sophisticated, non-incremental optimization algorithms. The only exception is the grid graph (A) where *neato* outperforms *arf*. For graphs C and D however, *arf* performs better. Please note that this benchmark only aims at small graphs. And the selection of only four graphs is not representative.

To address the points "separation of clusters" and "usage of available space", Fig. 3 shows an overview of layouts produced by *arf* and the *graphviz* layouters. The layouters were all called with the same input dot-file and without any further

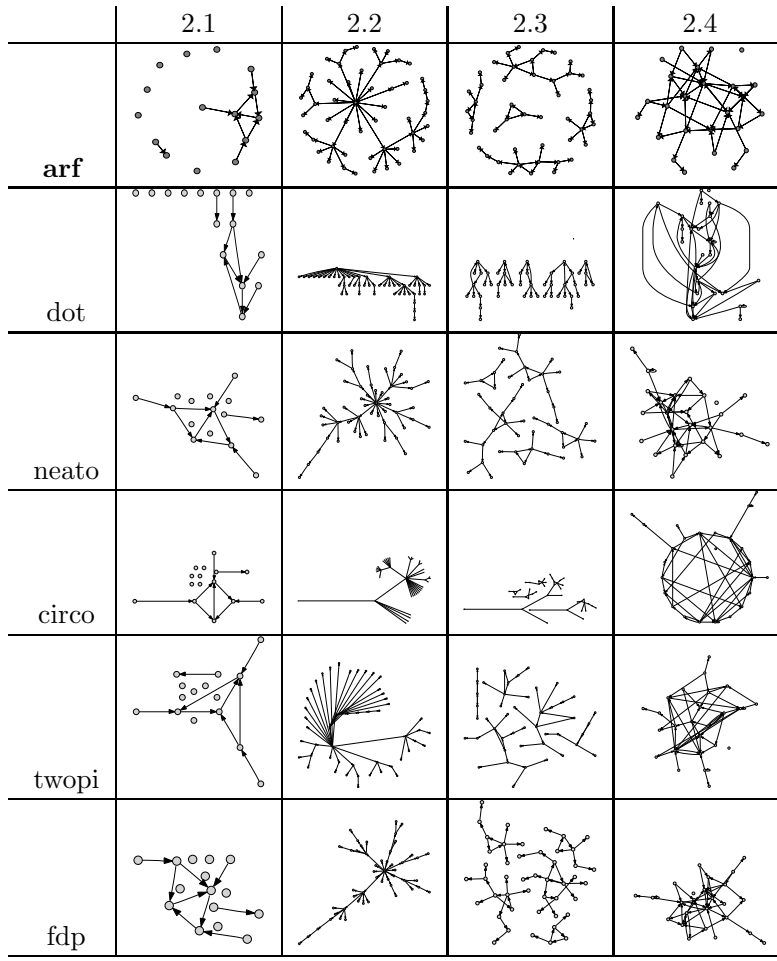


Fig. 3. Four different networks were visualized by various layouters: Arf, dot, neato, circo, twopi and fdp. The graphs can be characterized as follows: E is a network with two connected components and many unconnected nodes. F shows a scalefree network. G is a set of dentritic clusters. Finally H shows a random graph.

options.[c] The first row shows the layout produced by *arf*. The others show the layout produced by the different layouters in the *graphviz* package. Right from the first graph (E), an advantage of *arf* can be seen: connected subgraphs are clearly separated, whereas in the layout produced by *neato*, *twopi*, and *fdp*, single nodes get "caught" in the "kite"-like subgraph. Looking at the layout of graph (F) — a scale free network — the second column clearly shows *arf*'s ability to use the layout space economically. Also for the dentritic clusters in graph (G), *arf* produces competitive

[c]To make the results reproducible, the dot-files are available at `http://www.sg.ethz.ch/research/` as is source code in Java.

results: The clusters are clearly separated and crossing free. Finally, *arf* produces a clearer and thus more readable layout for a random network (2.4). The nodes are more clearly arranged.

5. Application Examples

This section is devoted to the practical application of *arf* in the field of complex systems research. The first examples deal with the visualization of social network diagrams and auto catalytic networks. Next we show how state-of-the-art animations can be generated in order to shed light on the dynamics of networks with changing structure. Finally, we highlight how the *arf* model can be fruitfully employed to explore large scale networks.

5.1. *Autocatalytic and social networks*

Crucial for interpreting networks is a visualization, that makes significant features easy to be perceived by the human eye. The following examples shall illustrate this.

Figure 4 shows an autocatalytic network, taken from Ref. 15. The network was laid out twice: The layout on the left is taken from the original publication (it was generated with *twopi*). The layout produced by *arf* (right) is far easier to read and to interpret: The network contains one big connected component, three two-node-components and numerous single nodes. The connected component can be divided into a core forming an autocatalytic cycle (marked red) and a dentritic periphery. All these characteristic features are clearly visible in the right visualization but not in the left one.

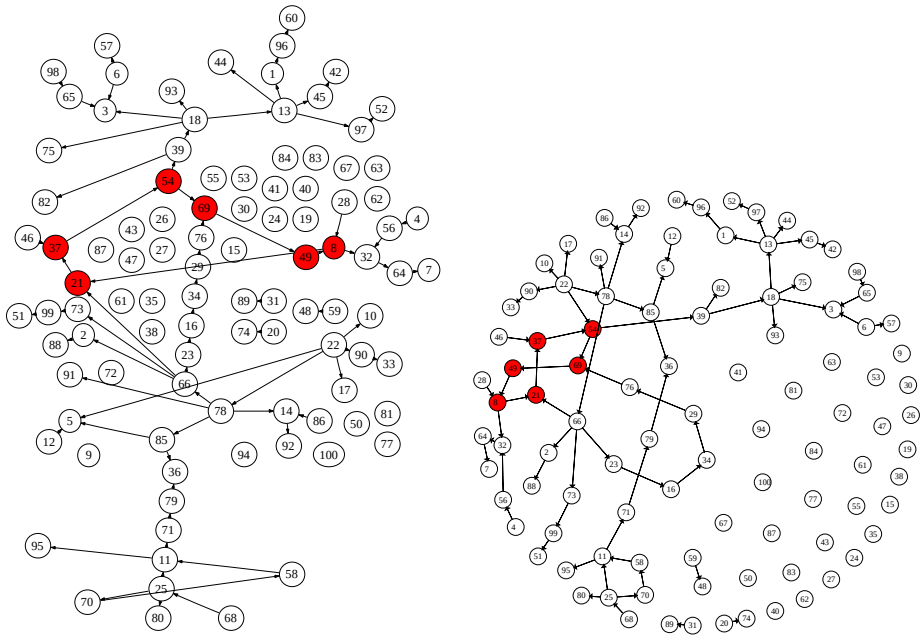
Next we revisit the social network presented in Sec. 1. In Fig. 1 managers of firms are connected to other board members by joint board membership.[d] Figures 1(a) and 1(b) are generated with *neato* and *fdp*. Figure 1(c) was generated with *arf*. Even though the big star takes more space in (c) than in (a) and (b), both the strongly connected core, as well as the periphery are given more space than in the other visualization. Their structure is clearer and it is easier to discern different cliques of nodes.

5.2. *Animating dynamic graphs*

In this section we describe how *arf* can be used to produce state-of-the-art animations. The basic scheme is shown in Algorithm (2). In the outer loop, the algorithm iterates through all the changes c in the set of changes C that are to be performed on the graph G. These changes can, for example, be codified in a script:

```
addnode 12, addedge 12 1;
dropedge 12 1;
```

[d]The data is taken from the ORBIS 07 database (Bureau van Dijk).



(a) (b)

Fig. 4. An auto catalytic network taken from, Ref. 15 laid out by *twopi* (a) and *arf* (b). *Arf* clearly shows the structural features of the graph: one big connected component with a dentritic periphery and a six-cycle (marked •).

Algorithm (2)

1: **for all** $c \in C$ **do**
2: perform c on G
3: **For** k **times do**
4: relayout G
5: render G
6: **end for**
7: **end for**

A similar script approach was also presented by Ref. 2 to codify and visualize changes in the WWW. This example would simultaneously add a node with id 12 and an edge from it to node 1. In the next graph change, the very same edge would be dropped. Now that the graph changed, the layout has to be adapted. This is done in k steps to ensure a smooth transition from layout to layout. The smaller k is, the faster the animation. The relayout procedure is just the body of the while-loop (lines 2–7) in Algorithm (1). After each relayout the graph is rendered to the screen or an image file, respectively.

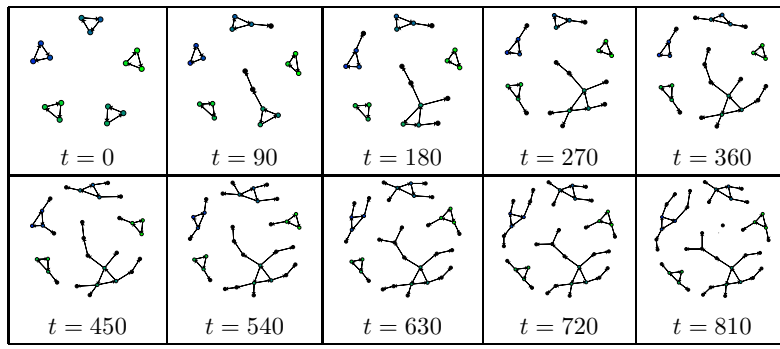


Fig. 5. A dynamically changing graph laid out with *arf*.

In the latter case, the generated image files can easily be compiled to a video file of any format by using openly available graphics tools. Figure 5 shows an example of such an animation. Exactly the same procedure was used in actual research practice: The simulation of R&D cooperation between firms was animated with *arf*.[10] The videos can be downloaded at `http://www.sg.ethz.ch/research/graphlayout/` as well as several other videos of evolving networks.

5.3. *Exploring large networks*

Often networks are too large to be visualized completely. Just imagine cross-investment networks spanning entire countries or continents, the World Wide Web or our civilization's social networks. In these cases, we can focus on only a small portion of the whole network at a given moment. The key concept is the one of a roving eye: There is one node in focus. Only this node and nodes within n degrees of separation are shown and laid out. Of course we want to shift the focus interactively, for example, by clicking on a node. The new node in focus defines a new set of visible nodes. Dynamic layout then gradually adjusts the positions of these nodes to generate a smooth transition from the old visualization to the new one.

Essential for the responsiveness of the interaction are the properties of the applied movement model. The *arf* model was successfully employed to perform this task. Large databases containing network data can easily be explored. Figure 6 shows a snapshot. In the main window a section of the ownership network of Swiss firms is shown.[e] On the right more detailed information on the focal node is presented.

6. Conclusion

This paper revisited the concept of force directed graph layout and put it into a complex systems context. An analysis of the classical spring model identified

[e]Based on the ORBIS 2007 Database (Bureau van Dijk).

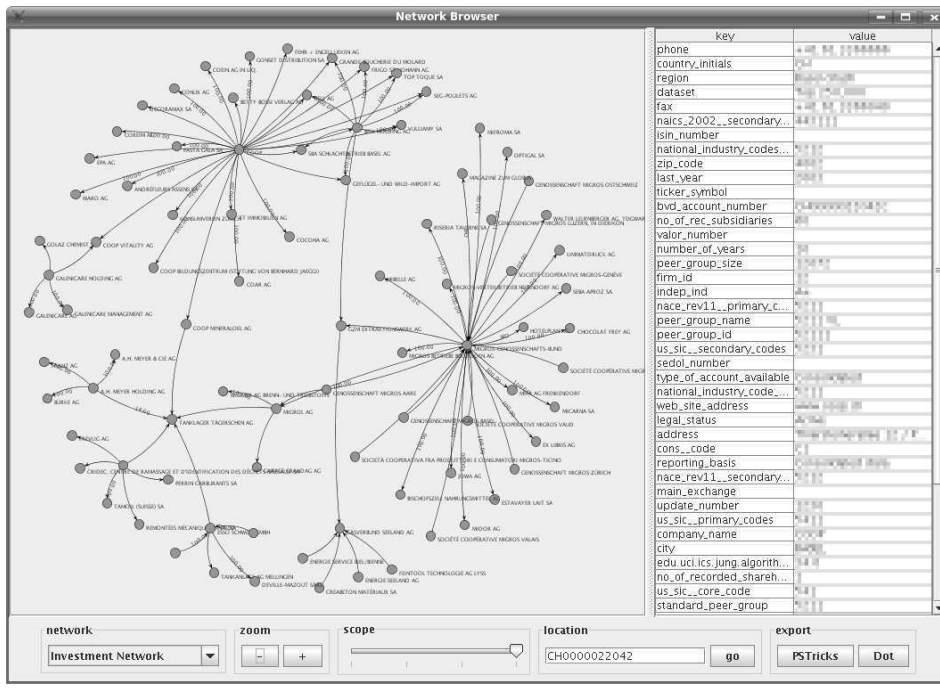


Fig. 6. A screen shot of the prototypical network browser. The left panel shows a part of the cross investment network of Swiss firms, surrounding the retailer "COOP". The right panel shows firm specific details.

several deficiencies of the resulting layout. A new model was derived to resolve them. A model, especially suited for dynamic layout and the visualization of small ($n < 10^3$) networks with strongly heterogeneous node degrees. The performance of the new model was evaluated and several successful applications in the field of complex systems research were discussed. Demonstrations and videos can be found at `http://www.sg.ethz.ch/research/graphlayout`. A prototypical open source implementation of the new layouter is also available. It can easily be plugged into the open source graph library *JUNG*.[f]

Acknowledgments

I express my gratitude to Frank Schweitzer (Zurich), Stefano Battiston (Zurich) and Frank Walter (Zurich) for their valuable comments and suggestions.

References

1. A.-L. Barabasi, *Linked: How Everything is Connected to Everything Else and What it Means* (Plume, 2003).

[f]Website: `http://jung.sourceforge.net/`. See also Ref. 13.

2. U. Brandes, V. Kääb, A. Löh, D. Wagner and T. Willhalm, Dynamic WWW structures in 3D, *J. Graph Algorithms Appl.* **4**(3), 183–191 (2000).
3. R. Davidson and D. Harel, Drawing graphs nicely using simulated annealing, *ACM Transactions on Graphics* **15**(4), 301–331 (1996).
4. P. Eades, A heuristic for graph drawing, *Congressus Numerantium* **42**, 149–160 (1984).
5. T. M. J. Fruchterman and E. M. Reingold, Graph drawing by force-directed placement, *Softw. Pract. Exper.* **21**(11), 1129–1164 (1991).
6. E. R. Gansner, E. Koutsofios, S. C. North and K.-P. Vo, A technique for drawing directed graphs, *Software Engineering* **19**(3), 214–230 (1993).
7. T. Kamada and S. Kawai, An algorithm for drawing general undirected graphs, *Inf. Process. Lett.* **31**(1), 7–15 (1989).
8. S. Kauffman, *At Home in the Universe: The Search for the Laws of Self-Organization and Complexity* (Oxford University Press, 1995).
9. M. Kaufmann and R. Wiese, Maintaining the mental map for circular drawings, in *Graph Drawing*, eds. M. T. Goodrich and S. G. Kobourov (Springer, 2002), pp. 12–22.
10. M. König, S. Battiston and F. Schweitzer, Microeconomic agent model of innovation networks, submitted, 2006.
11. R. Mach and F. Schweitzer, Modeling vortex swarming in daphnia, *Bulletin of Mathematical Biology* **69**, 539 (2007).
12. H. Meinhard, *The Algorithmic Beauty of Sea Shell* (Springer, 2003).
13. J. O'Madadhain, D. Fisher, P. Smyth, S. White and Y.-B. Boey, Analysis and visualization of network data using JUNG, *Journal of Statistical Software* (2005).
14. A. Quigley and P. Eades, FADE: Graph drawing, clustering, and visual abstraction, in *GDRAWING: Conference on Graph Drawing (GD)* (2000).
15. A. Seufert and F. Schweitzer, Aggregate dynamics in an evolutionary network model, *Int. J. Mod. Phys. C* (2006).
16. J. M. Six and I. G. Tollis, A framework for circular drawings of networks, in *GDRAWING: Conference on Graph Drawing (GD)*, ed. Kratochv (Springer, 1999), pp. 107–116.
17. S. H. Strogatz, Exploring complex networks, *Nature* **410**, 268–276 (2001).
18. M. Veloso, P. Stone and M. Bowling, Anticipation as a key for collaboration in a team of agents: A case study in robotic soccer, in *Proceedings of SPIE Sensor Fusion and Decentralized Control in Robotic Systems II*, Vol. 3839, eds. P. S. Schenker and G. T. McKee, Bellingham, WA, September 1999 (SPIE), pp. 134–143.
19. G. J. Wills, Nicheworks — interactive visualization of very large graphs, in *GD'97: Proceedings of the 5th International Symposium on Graph Drawing*, London, UK, 1997 (Springer-Verlag), pp. 403–414.