Dynamic Graph Drawing in Visone

Michael Baur and Thomas Schank*

April 22, 2008

Abstract

In early 2008 a new testing branch of the network analysis software visone has been made publicly available [Baur et al., 2008]. This realease includes the ability to process dynamic networks. A central feature is a dedicated dynamic layout algorithm which we present here from a methological and application oriented point of view.

We first describe this algorithm from an algorithmic perspective. More precisely we show how the local majorant method of the related layout algorithm given by Kamada and Kawai can be adapted to layout dynamic networks.

Following the methodological part we also address some problems and some other features of our implementation. Given the information in this document users of the dynamic layout procedure in visone should be able to employ the various available options efficiently to get the desired results.

1 Introduction

Visualization of dynamic networks can be achieved by a sequence of static drawings, possibly with many of those such that they can be shown as an animation e.g. in a movie. As well as in the visualization of static networks achieving a pleasant drawing [Di Battista et al., 1999; Kaufmann and Wagner, 2001] is one of the key problems. Additionally the preservation of the mental map between successive layouts has been identified as the crucial objective in dynamic graph drawing [Misue et al., 1995]. How this can be realized has been fundamentally discussed e.g. in [Brandes and Wagner, 1997] and [Friedrich and Eades, 2002]. In the following several papers have been published which detail, improve, and moreover also present a reference implementation or a software that produces an animation for a specific instance e.g. [Brandes et al., 2000; Erten et al., 2004].

*`schank@ira.uka.de`

Software that can visualize and animate general dynamic networks in a productive environments has been introduced rather recently, for example in SoNIA [Moody et al., 2005] or has been added to existing network analysis software, e.g. in the case of PAJEK [Batagelj and Mrvar, 1998; de Nooy et al., 2004]. The preservation of the mental map relies in both cases on feeding an initial layout to an iterative method, which is expected to produce an result that is not far from the initial. We go one step further by adding a layout method for dynamic networks with inherent stability to the network analysis tool visone.

The rest of this work is organized as follows. Section 2 contains basic definitions and declarations. The core of this work is presented in Section 3 where the dynamic layout method is described. This is followed by some details and additional remarks with respect to the actual implementation in Section 4.

2 Preliminaries

2.1 Aggregation Graph

A undirected *graph* or in context of dynamic graphs the *aggregation graph*

$$G^a = (V^a, E^a)$$

consists of a set of Nodes V^a and a set of Edges $E^a \subset \{\{u, w\} : u, w \in V\}$, see for example Figure 1. We define $n = |V^a|$ and $m = |E^a|$ and assume that the nodes are indexed from 1 to n. We will not distinguish between the elements of V^a, denoted by $u, v, w, \ldots$ and the indexes, denoted by $i, j, \ldots$ in the following.

2.2 Temporal or Dynamic Graph

A *dynamic graph* consists of the aggregation graph, an interval $T = \{1, 2, \ldots, N\}$ in $\mathbb{N}$, and two functions

$$\delta_V : V^a \times T \to \{0, 1\}$$

$$\delta_E : E^a \times T \to \{0, 1\}$$

with the meaning that an node exists at time $t \in T$ if and only if $\delta_V(u, t) = 1$. The existence of edges is defined and interpreted analogously. For $t \in T$ we require $\delta(\{u, w\}, t) = 0$ if $\delta(u, t) = 0$ or $\delta(w, t) = 0$, i.e. an edge can only exist if both endpoints exist. In the following we simply use δ for δ_V or δ_E.

We can now define a time dependent set of nodes $V_t = \{v \in V : \delta(v, t) = 1\}$ and a time dependent set of edges E_t analogously. The time dependent graph is finally defined as

$$G = \{G_t = \{V_t, E_t\} : t \in T\}.$$

2.3 Layouts

A *(static) layout* is mapping of the nodes to $\mathbb{R}^d$

$$\mathbf{x} : V \to \mathbb{R}^d$$

.

In a (straight line) *drawing* the nodes are placed according to the layout in the plane, hence $d = 2$ usually. An edge $\{u, w\}$ is drawn as a straight line segment between the points $\mathbf{x}(u)$ and $\mathbf{x}(w)$. Quality measures and computation of layouts are discussed in [Di Battista et al., 1999] and [Kaufmann and Wagner, 2001] for example.

A *dynamic layout* is a series of $|T|$ static layouts

$$\mathbf{x}_t : V_t \to \mathbb{R}^d.$$

Besides the objectives of a static layout it is required that the *mental map* between two consecutive layouts $\mathbf{x}(t)$ and $\mathbf{x}(t+1)$ does not "differ much", such that the observer can easily identify corresponding structures [Misue et al., 1995]. See [Brandes and Wagner, 1997; Branke, 1999; Diehl and Görg, 2002] for a more extensive discussion of dynamic layouts.

3 The Dynamic Layouter

As mentioned various methods exist to compute layouts. A popular principle is to associate the layout with a *energy* or *stress* and iteratively relayout such that this energy becomes lesser. This principle has been used in different variations, e.g. analogously to physical springs [Fruchterman and Reingold, 1991], or by gradient decent methods of an energy defined by graph distances [Kamada and Kawai, 1989]. See [Brandes, 1999] for an overview.

In the following we will use an adaptation of an energy minimizing method to compute dynamic layouts. We start with a short review of the original algorithm as it is used for static graphs before discussing how to adapt it to layout dynamic networks.

3.1 Static Layouts by Local Majorant Minimization

The layout method of Kamada and Kawai can be seen as a special version of the more general concept of multidimensional scaling. The target distances d_{ij} for each pair of nodes is usually set to the graph theoretic distance $dist_{ij}$, i.e. the length of the shortest path between i and j. Kamada and Kawai associate each target distance d_{ij} additionally with a weight w_{ij} indicating how important it is to obey the distance d_{ij} in the final layout. The weight is usually set to $w_{ij} = dist_{ij}^{-2}$.

For a layout $\mathbf{x}$ the energy or stress between a pair of nodes is then defined as

$$s_{ij} = w_{ij} \left(d_{ij} - ||\mathbf{x}_i - \mathbf{x}_j|| \right)^2 . \tag{1}$$

Expressed in words it is the square of the difference between actual distance in the layout $||\mathbf{x}_i - \mathbf{x}_j||$ and the target distance d_{ij} multiplied by the weight or importances w_{ij} of obeying this target distance. The total stress of the layout is then given by summing up the stress of all pairs of nodes

$$S = \sum_{j \neq i} d_{ij}^{-2} \left(d_{ij} - ||\mathbf{x}_i - \mathbf{x}_j|| \right)^2 . \tag{2}$$

Kamada and Kawai use a gradient decent method to iteratively minimize the stress according to Equation 2 and compute the layout. Gansner, Koren, and North propose to minimize a majorant of S instead. The authors show that their method has several benefits. The majorant can be minimized efficiently by matrix methods and is less likely to get stuck in local minima [Gansner et al., 2004].

Additionally this minimization can be performed by a simple localized algorithm which can be easily modified. The localized method moves a node i in dimension d to its new location according to

$$\text{new-}x_i^{(d)} = \frac{\sum_{j \neq i} w_{ij} \left(x_j^{(d)} + d_{ij} \frac{x_i^{(d)} - x_j^{(d)}}{||\mathbf{x}_i - \mathbf{x}_j||} \right)}{\sum_{j \neq i} w_{ij}} . \tag{3}$$

An iterative layout algorithm can be constructed accordingly:

```
repeat
   for each node i
      in each dimension d
            $x_i^{(d)} \leftarrow \text{new-}x_i^{(d)}$
until the total stress is minimized appropriatly
```

The outer loop is usually repeated for a fixed number of steps or until the stress does not improve by more than a small fraction. If we assume that it is carried out for a fixed number of steps the algorithm runs in $\mathcal{O}(n^2)$ time.

3.2 Dynamic Layouts by Local Majorant Minimization

We extend Equations 1, 2, and 3 to relate positions of $\mathbf{x}_{i,t}$ with the position of its predecessor $\mathbf{x}_{i,t-1}$ and successor $\mathbf{x}_{i,t+1}$. Thereby we limit the distance of those and provide stability to consecutive layouts.

For a dynamic graph G let $d_{ij,t}$ and be w_{ij} analogously defined as before. An additional parameter $\omega \in \mathbb{R}_{\geq 0}$ influences the stability of of the dynamic layout. For $\omega = 0$ the layout for each time step is independent from the previous or following layout. For $\omega > 0$ the position $\mathbf{x}_{i,t}$ is related to $\mathbf{x}_{i,t-1}$ by adding terms to the corresponding equations with target distance zero.

The stress for a pair of nodes within V_t is equivalent to the stress for a pair of nodes in the static case (Equation 1):

$$s_{ij,t} = w_{ij} \left(d_{ij,t} - ||\mathbf{x}_{i,t} - \mathbf{x}_{j,t}|| \right)^2 . \tag{4}$$

The total stress consists of the stress of all pairs in V_t summed up over all steps of time, and the stress arising from stability which is reflected in the second term of the following equation

$$S = \left[\sum_t \sum_{i \neq j} s_{ij,t} \right] + \left[\sum_{1 \leq t < |T|} \delta_{i,t} \; \delta_{i,t+1} \; \omega \; \; ||\mathbf{x}_{i,t} - \mathbf{x}_{i,t+1}||^2 \right] . \tag{5}$$

The iteratively computed new coordinate for $x_{i,t}$ is accordingly

$$\text{new-}x_{i,t}^{(d)} = \frac{\left[\sum\limits_{j \neq i} w_{ij} \left(x_{j,t}^{(d)} + d_{ij,t} \frac{x_{i,t}^{(d)} - x_{j,t}^{(d)}}{||\mathbf{x}_{i,t} - \mathbf{x}_{j,t}||} \right) \right] + \omega(\delta_{i,t-1} x_{i,t-1}^{(d)} + \delta_{i,t+1} x_{i,t+1}^{(d)})}{\left[\sum_{j \neq i} w_{ij} \right] + \omega(\delta_{i,t-1} + \delta_{i,t+1})} . \tag{6}$$

The δ-Functions in Equation 5, and 6 can be replaced by factor 1 instead (Equation 6 has to be adopted slightly) in this case stability will be provided if a node disappears and reappears some time later.

The derived algorithm would look like the following:

```
repeat
   for each time t
      for each node i
         in each dimension d

                    $x_{i,t}^{(d)} \leftarrow \text{new-}x_{i,t}^{(d)}$

until the total stress is minimized appropriately
```

With the same assumptions as previously the running time is in $\mathcal{O}(n^2 \cdot |T|)$. Note, that we disregard the costs of computing the values $d_{ij,t}$ here.

Conclusion 1 *The proposed dynamic layout algorithm does not impose a larger asymptotic running time as compared to computing the static layouts for each step of time separately.*

4 Details and Options in the Implementation

We list some of details of our implementation here that can be influenced by options in the dialog of the layouter. Figure 5 shows a screenshot of this dialog.

4.1 Stability

The value for the stability ω as in Equation 5 respectively 6 can be set. Higher values of ω provide more stability. Setting $\omega = 0$ is equivalent to compute the layout for each step of time independently.

4.2 Improving the Resolution by Edge Length Variation

By replacing the uniform edge lengths in computing standard graph theoretic distances by other values it is possible to improve some aspects of the layout.

If the sized of the nodes vary largely, e.g. due to a analysis visualization, it is possible add the radii of both end nodes to the target edge length.

Stepping further in this direction dense regions of the network can be expanded, see Figure 7 for an example. This is done by considering also the sizes of adjacent nodes for each endpoint.

4.3 Influence of the Initial Layout

As our methods improves a given layout iteratively the quality of the final layout, the stability, and also the number of required iterations to achieve a certain quality and hence the execution time can depend on the initial layout.

In cases where the dynamic network does only change little over time it may suffice to iterate from a common layout of the aggregation in order to provide stability even if ω is set to zero. The description in [Moody et al., 2005] is not very clear. However, it suggest that such an approach alone is used to provide stability there.

In practice we find that the quality and final stress seems not to depend drastically on the initial layout. However the number of iterations can increase significantly for example when using independent random layouts for each G_t.

The default in the current implementation is to use the coordinates from the layouted aggregation graph as an initial values. This layout of the aggregation itself is computed first by *Pivot-MDS* [Brandes and Pich, 2007] followed by static stress minimization as discussed in Section 3.1. We find that the number of required iterations is reduced efficiently. Furthermore, the procedure has also the benefit of being deterministic.

4.4 Handling Disconnected Components and Separation

The method of Kamada and Kawai and its derivations does not work on disconnected graphs without modifications. If G_t is disconnected some of the target distances $d_{ij,t}$ will become infinity in the distance computation.

In the current implementation we substitute these values by the maximum non infinite distances in G_t

$$d_{\max,t} = \max_{i \neq j}\{d_{ij,t} : d_{ij,t} \neq \infty\}$$

or optionally by the maximum over all of those

$$d_{\max} = \max_{t}\{d_{\max,t}\}.$$

However, we found that both of these options tend to separate the components more than desired. To ameliorate this $d_{\max,t}$ or $d_{\max}$ can be optionally scaled by an factor between 0 and 1.

Even though we are able to produce pleasant drawings of dynamic networks with disconnected components, we have to recognize that finding the appropiate values requires experience as well as iterative manual procedure and is therefore not optimal. We consider it as an open problem to find a better solution that will give good results without the need of setting parameters.

4.5 Termination

The number of iterations can be limited by a constant factor. Additionally it is also possible to set a threshold that terminates if a “normalized” modification of the stress given by Equation 5 does not improve by more as this threshold from the previous to the current iteration.

References

Vladimir Batagelj and Andrej Mrvar. Pajek – program for large network analysis, 1998.

Michael Baur, Ulrik Brandes, Thomas Schank, and Dorothea Wagner. Dynamic network analysis and visualization with visone. In *401. Wilhelm and Else Heraeus Seminar: Evolution and Physics Concepts, Models and Applications*, 2008. poster.

Ulrik Brandes. Drawing on physical analogies. In Kaufmann and Wagner [2001], pages 71–86. ISBN 3-540-42062-2.

Ulrik Brandes and Christian Pich. Eigensolver methods for progressive multidimensional scaling of large data. In Michael Kaufmann and Dorothea Wagner, editors, *Graph Drawing, Karlsruhe, Germany, September 18-20, 2006*, pages pp. 42–53. Springer, 2007.

Ulrik Brandes and Dorothea Wagner. A Bayesian paradigma for dynamic graph layout. In Giuseppe Di Battista, editor, *Proceedings of the 5th Symposium on Graph Drawing (GD'97)*, volume 1353 of *Lecture Notes in Computer Science*, pages 236–247, Rome, Italy, 18–20 September 1997. Springer-Verlag.

Ulrik Brandes, Vanessa Kääb, Andres Löh, Dorothea Wagner, and Thomas Willhalm. Dynamic www structures in 3d. *J. Graph Algorithms Appl.*, 4(3): 183–191, 2000.

Jürgen Branke. Dynamic graph drawing. In Kaufmann and Wagner [2001], pages 228–246. ISBN 3-540-42062-2.

Wouter de Nooy, Andrej Mrvar, and Vladimir Batagelj. *Exploratory Social Network Analysis with Pajek*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521602629.

G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing*. Prentice Hall, Upper Saddle River, NJ, 1999.

Stephan Diehl and Carsten Görg. Graphs, they are changing -dynamic graph drawing for a sequence of graphs. In Michael T. Goodrich and Stephen G. Kobourov, editors, *Graph Drawing, Irvine, CA, USA, August 26-28, 2002*, pages pp. 23–30. Springer, 2002.

Cesim Erten, Philipp J. Harding, Stephen G. Kobourov, Kevin Wampler, and Gary V. Yee. Graphael: Graph animations with evolving layouts. In Giuseppe Liotta, editor, *Graph Drawing, Perugia, Italy, September 21-24, 2003*, pages 98–110. Springer, 2004.

Carsten Friedrich and Peter Eades. Graph drawing in motion. *J. Graph Algorithms Appl.*, 6(3):353–370, 2002.

Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Softw., Pract. Exper.*, 21(11):1129–1164, 1991.

Emden R. Gansner, Yehuda Koren, and Stephen C. North. Graph drawing by stress majorization. In János Pach, editor, *Graph Drawing*, volume 3383 of *Lecture Notes in Computer Science*, pages 239–250. Springer, 2004. ISBN 3-540-24528-6.

T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Inf. Process. Lett.*, 31(1):7–15, 1989. ISSN 0020-0190.

Michael Kaufmann and Dorothea Wagner, editors. *Drawing Graphs, Methods and Models (the book grow out of a Dagstuhl Seminar, April 1999).*, volume 2025 of *Lecture Notes in Computer Science*, 2001. Springer. ISBN 3-540-42062-2.

Kazuo Misue, Peter Eades, Wei Lai, and Kozo Sugiyama. Layout adjustment and the mental map. *J. Visual Languages and Computing*, 6(2):183–210, June 1995. ISSN 1045-926X.

J. Moody, D. Mcfarland, and Bender S. Demoll. Dynamic network visualization. *American Journal of Sociology*, 110(4):1206–41, 2005.

A Figures

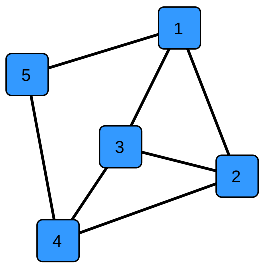
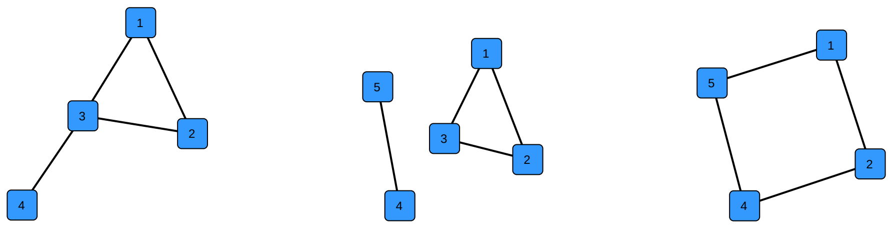
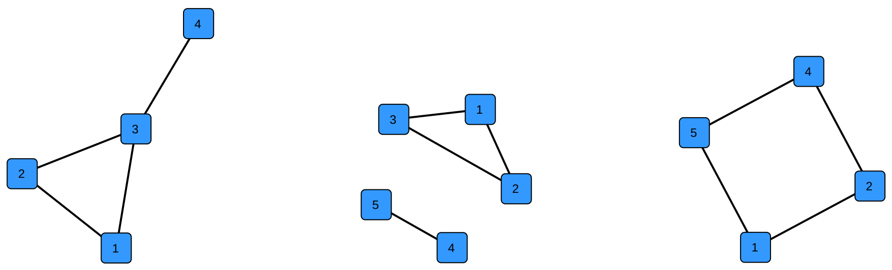


Figure 1: Aggregation Graph



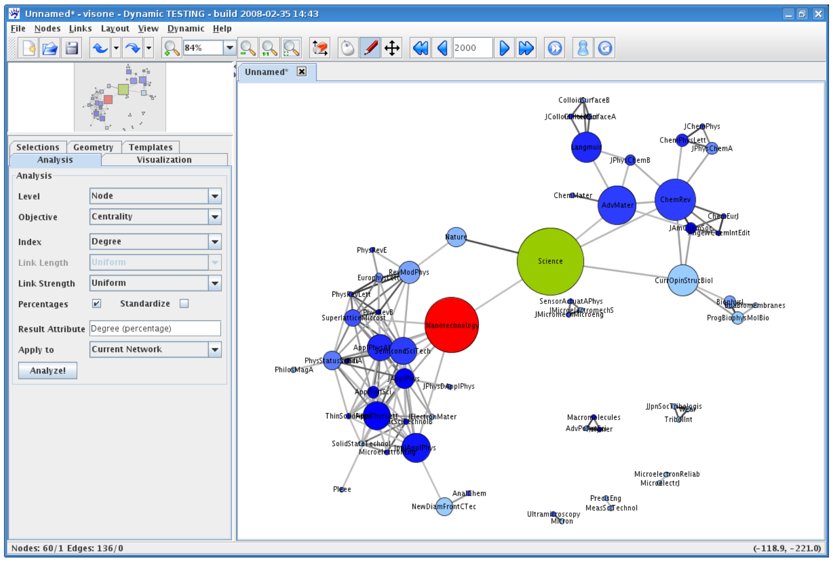
(a) $G_{t=1}$ (b) $G_{t=2}$ (c) $G_{t=3}$

Figure 2: Dynamic Graph



(a) $G_{t=1}$ (b) $G_{t=2}$ (c) $G_{t=3}$

Figure 3: Independent Layouts of the Dynamic Graph

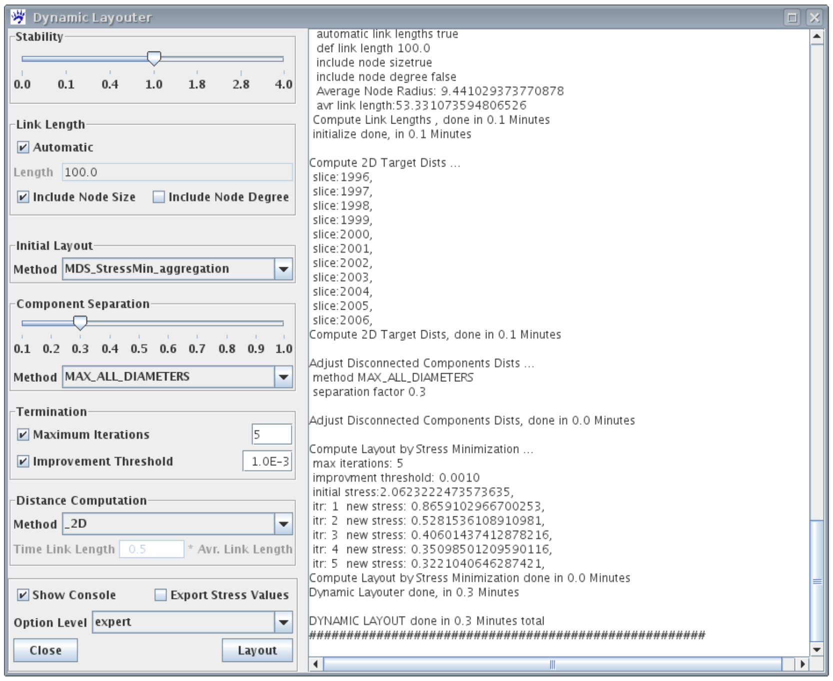

Unnamed* - visone - Dynamic TESTING - build 2008-02-35 14:43
File Nodes Links Layout View Dynamic Help
Selections Geometry Templates
Analysis Visualization
Unnamed*

Analysis

Level: Node
Objective: Centrality
Index: Degree
Link Length: Uniform
Link Strength: Uniform
Percentages ☑ Standardize ☐
Result Attribute: Degree (percentage)
Apply to: Current Network
Analyze!

Nodes: 60/1 Edges: 136/0 (-118.9, -221.0)

Figure 4: Visone Window



Stability: 0.0 0.1 0.4 1.0 1.8 2.8 4.0

Link Length
☑ Automatic
Length 100.0
☑ Include Node Size ☐ Include Node Degree

Initial Layout
Method MDS_StressMin_aggregation

Component Separation: 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
Method MAX_ALL_DIAMETERS

Termination
☑ Maximum Iterations 5
☑ Improvement Threshold 1.0E-3

Distance Computation
Method _2D
Time Link Length 0.5 * Avr. Link Length

☑ Show Console ☐ Export Stress Values
Option Level expert
Close Layout

```
 automatic link lengths true
 def link length 100.0
 include node sizetrue
 include node degree false
 Average Node Radius: 9.441029373770878
 avr link length:53.331073594806526
Compute Link Lengths , done in 0.1 Minutes
initialize done, in 0.1 Minutes

Compute 2D Target Dists ...
 slice:1996,
 slice:1997,
 slice:1998,
 slice:1999,
 slice:2000,
 slice:2001,
 slice:2002,
 slice:2003,
 slice:2004,
 slice:2005,
 slice:2006,
Compute 2D Target Dists, done in 0.1 Minutes

Adjust Disconnected Components Dists ...
 method MAX_ALL_DIAMETERS
 separation factor 0.3

Adjust Disconnected Components Dists, done in 0.0 Minutes

Compute Layout by Stress Minimization ...
 max iterations: 5
 improvment threshold: 0.0010
 initial stress:2.0623222473573635,
 itr: 1  new stress: 0.8659102966700253,
 itr: 2  new stress: 0.5281536108910981,
 itr: 3  new stress: 0.4060143741287821 6,
 itr: 4  new stress: 0.35098501209590116,
 itr: 5  new stress: 0.322104064628742 1,
Compute Layout by Stress Minimization done in 0.0 Minutes
Dynamic Layouter done, in 0.3 Minutes

DYNAMIC LAYOUT done in 0.3 Minutes total
#######################################################
```

Figure 5: Dialog of the Dynamic-Layouter

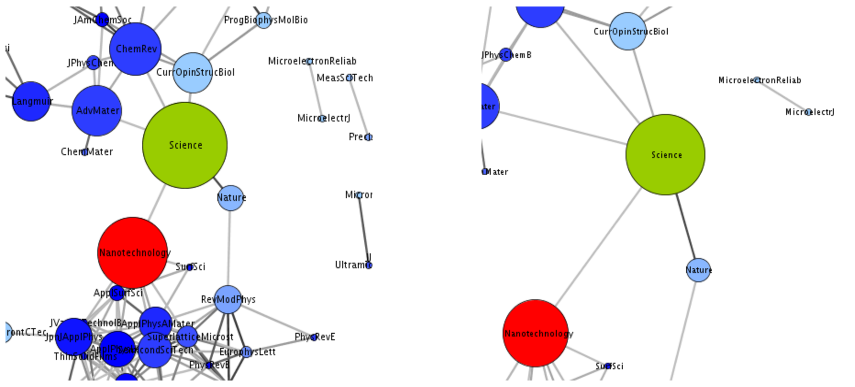


Figure 6: Include Size of Nodes

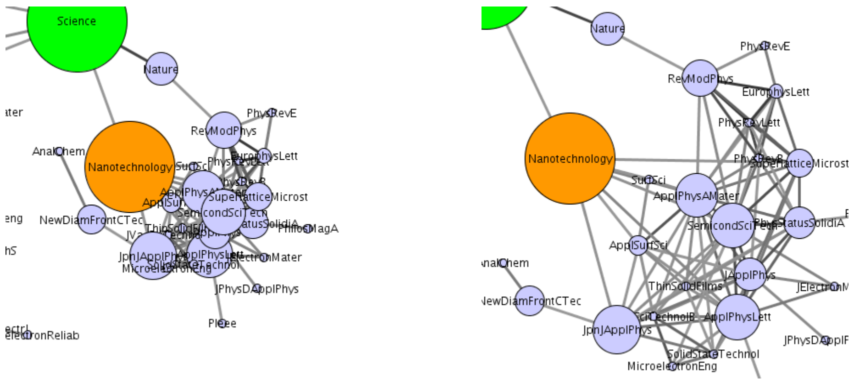
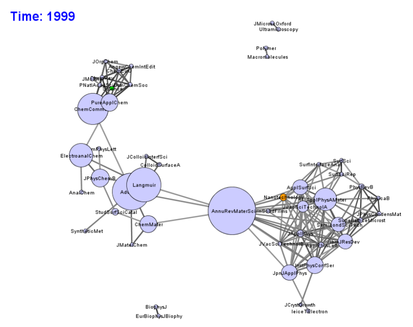
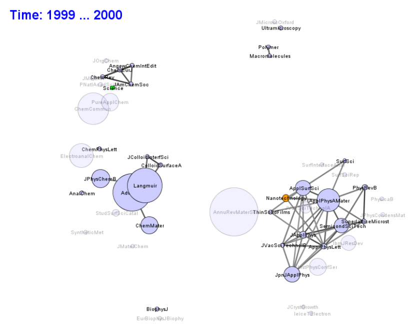


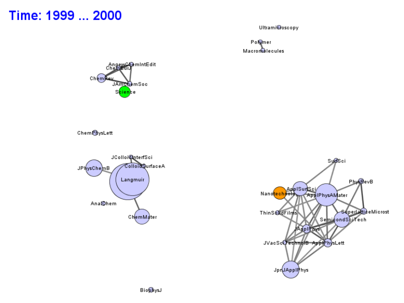
Figure 7: Expanding dense Regions



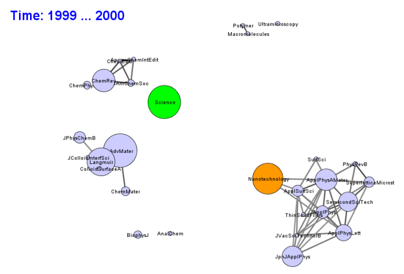
(a) Start



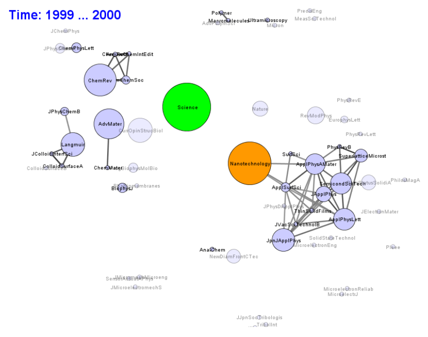
(b) Fading-out



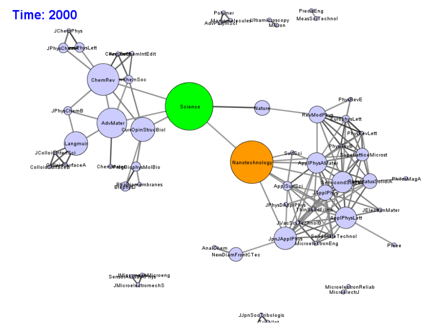
(c) Moving and Morphing



(d) Moving and Morphing



(e) Fading-in



(f) End

Figure 8: Snapshots of an Animation