Graphs, They Are Changing
Dynamic Graph Drawing for a Sequence of Graphs

Stephan Diehl and Carsten Görg

University of Saarland, FR 6.2 Informatik,
PO Box 15 11 50, D-66041 Saarbrücken, Germany
diehl@acm.org, goerg@cs.uni-sb.de

Abstract. In this paper we present a generic algorithm for drawing sequences of graphs. This algorithm works for different layout algorithms and related metrics and adjustment strategies. It differs from previous work on dynamic graph drawing in that it considers all graphs in the sequence (offline) instead of just the previous ones (online) when computing the layout for each graph of the sequence. We introduce several general adjustment strategies and give examples of these strategies in the context of force-directed graph layout. Finally some results from our first prototype implementation are discussed.

1 Introduction

Optimizing the layout adjustment for a sequence of graph changes, e.g. for an off-line animation, is still an open yet very challenging area of research. – *Jürgen Branke [3]*

Dynamic graph drawing addresses the problem of computing layouts of graphs which evolve over time by adding and deleting edges and nodes. This results in an additional aesthetic criterion known as "preserving the mental map" [13] or dynamic stability.

The ad-hoc approach is to compute a new layout for the whole graph after each update using those algorithms developed for static graph layout. In most cases this approach produces layouts which do not preserve the mental map. The common solution is to apply a technique known from key-frame animations called inbetweening to achieve "smooth" transitions between subsequent graphs, i.e. animations show how nodes are moved to their new positions. This approach yields decent results if only a few nodes change their position or whole clusters are moved without substantially changing their inner layout or if the transition is subdivided into several phases [9]. But in most cases the animations are just nice and do neither convey much information nor help to preserve the mental map.

2 Mental Map and Mental Distance

A layout is a mapping of the nodes and edges of a graph onto a plane or into a volume.

M.T. Goodrich and S.G. Kobourov (Eds.): GD 2002, LNCS 2528, pp. 23–31, 2002.

The term *mental map* refers to the abstract structural information a user forms by looking at the layout of a graph. The mental map facilitates navigation in the graph or comparison of it and other graphs. In the context of dynamic graph drawing changes to this map should be minimal, in other words algorithms to draw sequences of graphs should preserve the mental map. Misue et. al. [13] discuss three models for the mental map

- Orthogonality: The horizontal and vertical order of nodes should stay the same when changing node positions.
- Proximity Relations: The distance to all or at least to all neighboring nodes should be about the same.
- Topology: The dual graph should be maintained.

Based on each of these models we can define metrics that indicate how close the mental maps of two given layouts are. For the purpose of this paper we call the resulting class of metrics *mental distances*. Let Layout be the set of all layouts, i.e. in the simplest case mappings from nodes to points in $\mathcal{R}^2$.

Definition 1 (Mental Distance). *Let $l_1, l_2 \in \mathsf{Layout}$ be two layouts. Then the function $\Delta : \mathsf{Layout} \times \mathsf{Layout} \to \mathcal{R}_0^+$ is a metric for how good l_2 preserves the mental map of l_1. In particular $\Delta(l_1, l_2) = 0$ means that l_1 and l_2 have the same mental map.*

Most metrics only compare the layouts for the common subgraph of the underlying graphs, but we could also have metrics which additionally consider the layout of those nodes and edges which are only present in one of the layouts.

2.1 Examples of Mental Distances

Many researchers proposed distance metrics, e.g. [12,4]. We give here two which we use in our current implementation. The metrics below are based on node positions only. There are other metrics which take clustering or edge routing into account. For an overview see for example [3].

Definition 2 (Euclidean and Orthogonal Mental Distance). *Let $l_1, l_2 \in \mathsf{Layout}$ be two layouts. Their Euclidean mental distance is defined as $\Delta_{\|}(l_1, l_2) = \sum_{v \in V_1 \cap V_2} \mathsf{dist}(l_1(v), l_2(v))$ where $\mathsf{dist}(p, q)$ yields the Euclidean distance of two points. Let $\mathsf{sgn}(x) \in \{-1, 0, +1\}$ yield the sign of a (real) number x. Their orthogonal mental distance is*

$$\Delta_{\perp}(l_1, l_2) = \sum_{v_1, v_2 \in V_1 \cap V_2} \begin{array}{l} |\mathsf{sgn}(l_1(v_1).x - l_1(v_2).x) - \mathsf{sgn}(l_2(v_1).x - l_2(v_2).x)| \\ +|\mathsf{sgn}(l_1(v_1).y - l_1(v_2).y) - \mathsf{sgn}(l_2(v_1).y - l_2(v_2).y)| \end{array}$$

3 The Offline Dynamic Graph Drawing Problem

Assume that the quality of a layout could be measured by a function Γ regarding aesthetic goals like compactness, even distribution of nodes, minimal number of edge crossings, etc [15]. Such formal criteria are the computational crutches to substitute real models of human cognition or simply taste.

Definition 3 (Layout Quality). *Let $l \in \mathsf{Layout}$ be a layout. Then the function $\Gamma : \mathsf{Layout} \to \mathcal{R}_0^+$ is a metric for the quality of a single layout. In particular $\Gamma(l) = 0$ means that l has minimal quality.*

Now we can state the problem of dynamic drawing of a sequence of graphs. In most applications the graphs in such a sequence will result from changes to the preceding graph and thus will share some nodes and edges.

Definition 4 (The Offline Dynamic Graph Drawing Problem). *Given a sequence of n graphs $g_1, \ldots, g_n$. Compute layouts $l_1, \ldots, l_n$ for these graphs such that*

1. $\overline{\Delta} = \sum_{1 \le i < n} \Delta(l_i, l_{i+1})$ *is minimal* *2.* $\overline{\Gamma} = \sum_{1 \le i \le n} \Gamma(l_i)$ *is maximal*

Unfortunately, the two optimization goals can not be achieved at the same time in general. For the online dynamic graph drawing problem $l_1, \ldots, l_{n-1}$ and g_n are given and l_n has to be computed.

3.1 Foresighted Layout without Tolerance

In a previous paper we introduced a dynamic graph drawing algorithm which we called *Foresighted Layout* [7].

Given a sequence of n graphs we compute a global layout which induces a layout for each of the n graphs. In the simplest case this global layout is just the layout of the supergraph of all graphs in the sequence. By forming temporal equivalence classes of nodes and edges Foresighted Layout can also produce more compact global layouts. In a GAP (graph animation partition) nodes with disjoint live times are grouped together. In an RGAP (reduced GAP) also edges with disjoint live times are represented by a single edge. We have proven the $\mathcal{NP}$-completeness of computing a minimal GAP respectively RGAP by reduction on the minimal graph coloring problem [6,10].

A unique feature of the above approach is that once they are drawn on the screen neither nodes nor the bends of edges change their positions in graphs subsequently drawn. This algorithm preserves the mental map in a trivial way by using the global layout, but does so at the cost of other aesthetic criteria.

Foresighted Layout does not work for all classes of graphs equally well, as it requires that the supergraph, GAP or RGAP of the graphs belongs to the same class as the individual graphs. For example, in general the supergraph of trees is not a tree, and the supergraph of planar graphs is not a planar graph.

3.2 Foresighted Layout with Tolerance

In this paper we extend the above method, such that it can trade aesthetic quality for dynamic stability and vice versa. To this end we use a tolerance value δ and allow such layouts for individual graphs in the sequence for which the mental distance to certain other graphs[1] is smaller than δ. As a result we can formulate a weaker problem.

[1] The strategies that we are going to present differ particularly with regard to what other graphs are chosen for comparison.

Definition 5 (The Tolerant Offline Dynamic Graph Drawing Problem). *Given a sequence of n graphs $g_1, \ldots, g_n$ and a tolerance value δ. Compute layouts $l_1, \ldots, l_n$ for these graphs such that*

1. $\Delta(l_i, l_{i+1}) < \delta$ *for all* $1 \leq i < n$ *2.* $\overline{\Gamma} = \sum_{1 \leq i \leq n} \Gamma(l_i)$ *is maximal*

As a simple corollary we get that $0 \leq \overline{\Delta} < n * \delta$. In general we can expect that $\overline{\Gamma}$ increases for larger values of δ. In other words a small δ enforces dynamic stability, while larger values increase local quality.

For efficiency reasons we do not try to compute an optimal solution, but we compute approximations. Algorithm 1 is generic in the sense that it works with different static layout algorithms and related metrics and adjustment strategies. We use the notation $l_{|g}$ to denote the layout which results from restricting l to the nodes and edges in g. In other words $l_{|g}$ is the layout for g induced by the layout l. We use uppercase letters for the global layout L and layouts L_i for graphs g_i induced by the global layout. For all other layout we use lowercase.

Algorithm 1. Foresighted Layout with tolerance δ

```
Compute global layout L for supergraph (resp. GAP or RGAP) of g_1, ..., g_n
for i := 1 to n do
  L_i := L_|g_i
  l_i := adjust(...) // Compute l_i by adjusting L_i
                     // using one of the strategies discussed in Section 3.3
  if i==1 then
    Draw graph g_1 using l_1
  else
    Draw graph g_i by morphing from l_{i-1} to l_i
  end if
end for
```

3.3 Layout Adjustment Strategies

Classical layout adjustment methods $\mathsf{adjust}(g_i, l_{i-1})$ draw a graph g_i by adapting the layout l_{i-1} of the preceding graph. Our adjustment strategies also take the global layout into account. We use the supergraph (resp. GAP or RGAP) as a rough abstraction of the whole sequence of graphs. In a sense it contains information about the whole future. In addition the strategies might consider the previous, next or all graphs in the sequence. Instead of the graph g_i these strategies get the layout L_i for the graph induced by the global layout and try to adjust it while regarding constraints on the mental distance to other layouts. If they can not fulfill the constraints these strategies yield the induced layout.

Strategy 1 (Independent Adjustment). $l_i = \mathsf{adjust}(L_i, \delta)$
This strategy tries to preserve the mental map by ensuring that $\Delta(L_i, l_i) < \delta$. As a result all graphs in the animation stay close to the global layout.

Strategy 2 (Predecessor Dependent Adjustment). $l_i = \mathsf{adjust}(L_i, l_{i-1}, \delta)$ This strategy differs from the above by requiring that the layout stays close to that of the preceding one, i.e. $\Delta(l_{i-1}, l_i) < \delta$. Note, that there is no constraint for adjusting L_1. As a result l_1 can be very far from the induced layout and this can have undesirable effects. The value of $\Delta(l_1, L_2)$ might get greater than δ and all adjustments to L_2 might not sufficiently reduce the mental distance. In this case the adjustment will fail and return the induced layout.

Strategy 3 (Context Dependent Adjustment). $l_i = \mathsf{adjust}(L_i, l_{i-1}, L_{i+1}, \delta)$ This strategy extends the previous one by enforcing that the layout stays close to both the preceding layout as well as the induced layout for the subsequent graph, i.e. $\Delta(l_{i-1}, l_i) < \delta$ and $\Delta(l_i, L_{i+1}) < \delta$. In particular, this strategy makes sure that l_1 stays close to L_2 and thus we do not run into the problem discussed above when adjusting L_2.

The above strategies try to adjust a layout as much as possible, before proceeding to the next one. The previous layout can thus impose too much restrictions on the next one and render adjustments impossible. The following strategy strives to evenly adjust all layouts in the sequence.

Strategy 4 (Simultaneous Adjustment). $(l_1, \ldots, l_n) = \mathsf{adjust}(L_1, \ldots, L_n, \delta)$ This strategy simultaneously adjusts the induced layouts $L_1, \ldots, L_n$ such that $\Delta(l_i, l_{i+1}) < \delta$ for all $1 \leq i < n$. An variant of the simultaneous adjustment strategy could also try to preserve the inertia of movements.

3.4 Layout Adjustment for Force-Directed Layout

We show how a simple spring embedder [8,2] can be modified to perform layout adjustment according to the above strategies. Unfortunately once this embedders have computed a layout that does not preserve the mental map further iterations do not resolve the problem. As a result we extended these embedders by simulated annealing. We assume a global temperature T which cools off after each iteration.

Algorithm 2. $\mathsf{adjust}(L_i, l_{i-1}, L_{i+1}, \delta)$ context dependent

```
l := L_i
for j := 1 to #Iterations do
  Compute forces for each node in l with global temperature T
  Compute new layout l' by applying forces to nodes in l
  if Δ(l_{i-1}, l') < δ and Δ(l', L_{i+1}) < δ then
    l := l'
  end if
  T := anneal(T, j)
end for
return l
```

Algorithm 2 follows immediately from the above general description of the strategies. In Algorithm 3 the iterations of the embedder are performed simultaneously on all layouts. After each step we check whether the mental distances of a layout and the layouts of its previous and next graphs are below the tolerance value. If not the layout is discarded and the layout of the previous iteration is used for the next iteration.

Algorithm 3. adjust$(L_1, \ldots, L_n, \delta)$ simultaneous

$(l_1, \ldots, l_n) := (L_1, \ldots, L_n)$
for $j := 1$ **to** #Iterations **do**
  for $i := 1$ **to** n **do**
    Compute forces for each node in l_i with global temperature T
    Compute new layout l' by applying forces to nodes in l_i
    if $\Delta(l_{i-1}, l') < \delta$ **and** $\Delta(l', l_{i+1}) < \delta$ **then**
      $l_i := l'$
    end if
    $T :=$ anneal(T, j)
  end for
end for
return $(l_1, \ldots, l_n)$

---

4 Examples

In Figure 1 navigation through word collocation graphs is shown, more precisely the sequence consists of the three graphs for the words `Lecture`, `Corbate`, `Award` and `Turing`. In the first column the supergraph layouts of the graphs are shown. The nodes for the mentioned words have always the same relative position (the three layouts have small mental distances), but the layouts are of poor quality. In the second column context dependent adjustment with orthogonal mental distance and $\delta = 2$ is shown. The layout quality is better, but the mental distance is larger, because the nodes `Lecture` and `Corbate` have changed their horizontal order. In the third column simultaneous adjustment with orthogonal mental distance and $\delta = 2$ is shown. The layout of the first graph is even more compact (because more iterations could be performed) and the mental distance is equal to that of context dependent adjustment.

We have done experiments with several animations including navigation in the genealogy of the Hohenzollerns (family of German emperors), navigation in word collocation graphs based on data from `www.wortschatz.uni-leipzig.de`, computation of a full lattice for a given set of pairs of integers, as well as several random animations of graphs and trees. Obviously, these animations are difficult to show in a paper. Therefore a Java applet with examples is available online at `http://www.cs.uni-sb.de/~diehl/ganimation/`.

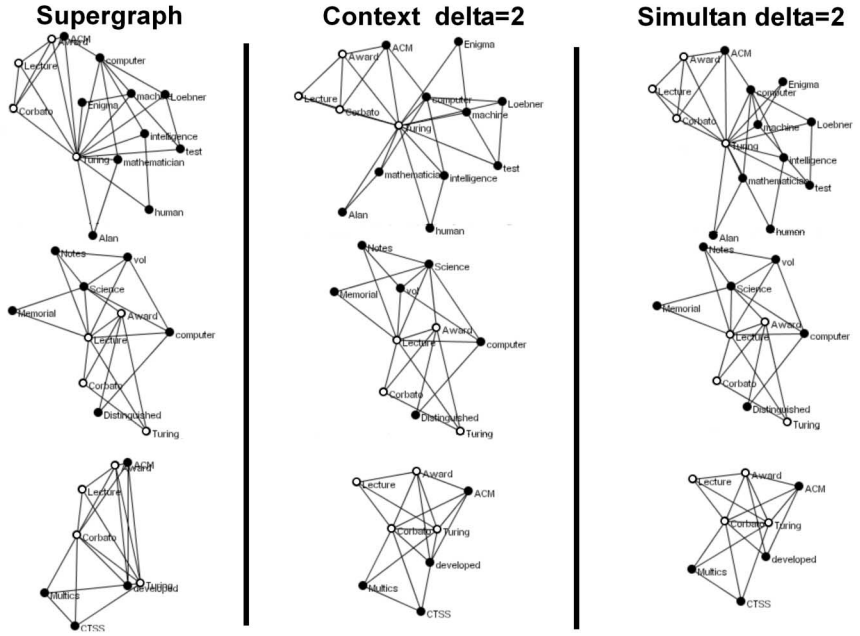

Fig. 1. Different adjustment strategies.

5 Implementation

In our first prototype implementation we implemented a spring embedder with polar and parallel magnetic fields, gravity, and simulated annealing similar to the algorithm used in VCG [16] (except that we do not use local temperatures). The prototype implements all strategies discussed in this paper and provides orthogonality and Euclidean distance as mental distances. Smooth transition between two graphs are animated in the following phases. First all deleted edges shrink and disappear. Then all deleted nodes disappear. Next all remaining nodes and edges are moved to their new positions using linear interpolation. Finally all new nodes appear and all new edges expand. We are currently working on adjustment strategies for hierarchical layout and these will be integrated into our original Foresighted Layout implementation that used hierarchical layout [7].

6 Related Work

So far, we have not found any published work on offline dynamic graph drawing. An overview of online dynamic graph drawing has e.g. been given in [3]. The general framework of Brandes and Wagner [1] characterizes the tradeoff between local quality and dynamic stability using conditional probabilities. In their formulation the conditional probability for a layout depends on those of the pre-

ceding ones, i.e. no look-ahead in the sequence is available, but the formulation could be easily adapted to the offline case.

Finally there are several papers that suggest layout adjustment algorithms for certain classes of graphs. All these algorithms have in common that they are based on a certain mental map distance built into the algorithm [13,5,12,14].

7 Conclusion and Future Work

We presented a generic algorithm to compute graph animations. We discussed several strategies for layout adjustment. The effectiveness of the resulting animations needs to be studied and evaluations similar to those of Purchase [15] for aesthetic rules must be performed for dynamic stability. For this we plan to implement additional layout algorithms and metrics.

The generic nature of our approach allows us to easily combine different layout algorithms, metrics and adjustment strategies. This comes at the cost of performance. One solution is to manually or automatically specialize the generic algorithm with respect to a given layout algorithm, adjustment strategy and metric before computing the layout. In a specialized algorithm the phases could be interleaved and allow more clever adjustment, e.g. only new coordinates are computed for those nodes which violate orthogonality.

References

1. U. Brandes and D. Wagner. A Bayesian paradigm for dynamic graph layout. In *Graph Drawing (Proc. GD '97)*, volume 1353 of *Lecture Notes Computer Science*. Springer-Verlag, 1997.
2. Ulrik Brandes. Drawing on physical analogies. In *Drawing Graphs [11]*. 2001.
3. Jürgen Branke. Dynamic graph drawing. In *Drawing Graphs [11]*. 2001.
4. S. Bridgeman and R. Tamassia. Difference metrics for interactive orthogonal graph drawing algorithms. In *Proceedings of 6th International Symposium on Graph Drawing GD'98*. Springer LNCS 1457, 1998.
5. R.F. Cohen, G. Di Battista, R. Tamassia, and I.G. Tollis. Dynamic graph drawings: Trees, series-parallel digraphs, and *st*-digraphs. *SIAM Journal on Computing*, 24(5), 1995.
6. S. Diehl, C. Görg, and A. Kerren. Foresighted Graphlayout. Technical Report A/02/2000, FR 6.2 - Informatik, University of Saarland, December 2000. http://www.cs.uni-sb.de/tr/FB14.
7. Stephan Diehl, Carsten Görg, and Andreas Kerren. Preserving the Mental Map using Foresighted Layout. In *Proceedings of Joint Eurographics – IEEE TCVG Symposium on Visualization VisSym'01*. Springer Verlag, 2001.
8. P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42, 1984.
9. C. Friedrich and M. E. Houle. Graph Drawing in Motion II. In *Proceedings of Graph Drawing 2001*. Springer LNCS (to appear), 2001.
10. M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of $\mathcal{NP}$-Completeness*. Freeman and Company, 1979.
11. M. Kaufmann and D. Wagner, editors. *Drawing Graphs – Methods and Models*, volume 2025 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.

12. K.A. Lyons, H. Meijer, and D. Rappaport. Cluster busting in anchored graph drawing. *Journal of Graph Algorithms and Applications*, 2(1), 1998.
13. K. Misue, P. Eades, W. Lai, and K. Sugiyama. Layout Adjustment and the Mental Map. *Journal of Visual Languages and Computing*, 6(2):183–210, 1995.
14. A. Papakostas and I.G. Tollis. Interactive orthogonal graph drawing. *IEEE Transactions on Computers*, 47(11), 1998.
15. H.C. Purchase, R.F. Cohen, and M. James. Validating graph drawing aesthetics. In F. J. Brandenburg, editor, *Graph Drawing (Proc. GD '95)*, volume 1027 of *Lecture Notes Computer Science*. Springer-Verlag, 1996.
16. G. Sander. *Visualization Techniques for Compiler Construction*. Dissertation (in german), University of Saarland, Saarbrücken (Germany), 1996.