Constraint–Based Spring–Model Algorithm for Graph Layout

Thomas Kamps and Joerg Kleinz
Institute for Integrated Publication and Information Systems (GMD–IPSI)
Dolivostr. 15, D–64293 Darmstadt, Germany, {kamps,kleinz}@darmstadt.gmd.de
John Read
Department of Combinatorics and Optimization
University of Waterloo, Ontario N2L 3G1, Canada, jread@orion.uwaterloo.ca

Abstract

In this paper we will discuss the question of how to develop an algorithm for automatically designing an optimal layout of a given constraint graph. This automatic layout algorithm must observe certain aesthetic principles, which facilitate the user's interpretation of the graph. The constraint graph is generated by the visualization algorithm AVE (Automatic Visualisation Engine) which decides in this case that the object relation between the objects which are to be graphically represented must be visually realized by lines. The constraints describe how subsets of objects are geometrically represented relative to each other. We require that each individual object be of fixed size throughout the algorithm, but we allow for each of these sizes to differ one from another. This automatic layout algorithm is developed along the lines of a (spring) force model, a method which has its roots in such works as [Eade]. As a measure of any particular layout's fidelity to our aesthetic principles, we have developed a function which assigns a real value to each possible layout. We have insured that this function has good differentiability properties, in order that we may exploit gradient descent methods to arrive at a layout that minimizes this function. Any such layout is then by definition the optimal layout we seek. These gradient methods are integral in assuring that the algorithm we develop can efficiently implement the aesthetic principles so that we obtain a very fast routine.

Introduction

In cooperation with Macmillan Publishers Ltd. the department PaVE (Publication and Visualisation Environment), as part of GMD–IPSI, has developed a prototype version of the electronic Dictionary of Art. In this application scenario an art historian, as a user of the interactive electronic refence work, queries a semantic network for information. The semantic network represents knowledge about the domain of art. The relations of that network establish facts between domain objects, e.g., "Frank Lloyd Wright's *profession* was architect" or "Applied Arts is a *broader term of* Graphic Design". In order to support the user in the process of information access we have built the system AVE (Automatic Visualisation Engine) that is able to automatically design diagrams as a visualisation of the query result. We refer the reader to [Reich] for detailed information concerning the generative theory underlying AVE's approach and to [Golo] for its implementation. In the following section we present a brief summary of AVE's features in order to explain the overall environment in which our spring–model algorithm is applied.

AVE's Approach

We follow a functional notion of aesthetics (see [Kamp94a]) and thus our approach to automatic data visualisation is based on the idea that the diagram should mirror the inherent characteristics of the data. As mentioned above we consider data to be represented as relations between objects. We interpret the relational properties as the characteristics, or the regularities, of the data and see the aggregations of properties, establishing the relation types, as a sensible way to type the data. To exploit data regularities is in analogy to many graph layout approaches. A bibliography of algorithms that graphically realize such properties ('tree', 'symmetric', 'hierarchy', et.) as line diagrams is given in [DiBa]. What makes the difference for us are requirements coming from our application. In the semantic network there are different sorts of relation types. Not all of them are sensibly visualized using the graphical resource 'line' as can be seen from the examples presented in figure 2. In order to integrate these different visulizations we think a rich classification of the data is inevitable. To describe these relations we use the Smalltalk Frame Kit (SFK) [Fisch], an object–oriented, frame–based representation language. SFK implements the theory of binary relations in order to describe the semantics of the relations in terms of their mathematical properties. We use the following incomplete but for our purposes sufficient classification schema for binary relations:

- Relation (unqualified)
  - Symmetric Relation (symmetric)
  - Acyclic Relation (acyclic)
    - Tree Relation (tree)
    - Irreflexive Order Relation (irreflexive, transitive, asymmetric)
      - Irreflexive Tree Order Relation (irreflexive–order, tree)
        - Discrete Linear Order Relation (irreflexive–tree–order, linear, discrete domain)
        - Continuous Linear Order Relation (irreflexive–tree–order, linear, cont. domain)
  - Bipartite Relation (bipartite)
    - Function (functional)

figure 1: classification schema for binary relations

Our philosophy is thereby to separate data and data analysis strictly from visualization decisions. [Kosa] proposed an approach in which they define subgraphs, e.g., 'T–Shapes' and 'Hub–Shapes' 'Axial Symmetry', etc., to be the regularities. We see these regularities not as an alternative to the ones we use but as an extension (a relation which is of order relation type may contain T–Shapes). However we would not make 'Axial Symmetry' a regularitiy because this is a purely geometric property and thus should become the result of a visualization decision. As a result of this discussion our aim is, similarly to that of [Kosa], to integrate results from the field of graph drawing but also form the field of information visualization for which we only representatively mention [Mack] and [Roth] who proposed approaches that mainly dealt with the visualization of functional relationships among data.
Using the above mentioned modeling language we can qualfiy relations with appropriate relation types on the class level. It depends on the application which types are assigned to which data relations. This is in accordance with the conformance criteria given by [Lin], which say that the visualization should be adaptable to the application.
In our approach we decompose the given semantic subnet into binary relations and analyse them to find out whether their types can be refined on the instance level. It might turn out that a relation which is defined to be of order relation type on the class level is of a more specific type, say tree order, on the given instance level. This strongly affects the visualization. We will explain this below. In order to decide how to assign graphical resources to the given relations we also rank them based on their specificity in the classification schema and based on the quantitative share that a binary relation has in the overall subnet. The first criterion is justified by the assumption that a specific relation type organizes the data more effectively than a more general type. For instance, a linear order relation organizes the data in a stronger way than an irreflexive order rela-

tion because in the case of linearity all elements are pairwise comparable whereas in the other case this is not true. The second criterion makes sure that a more specific relation type does not get the best graphical resources a priori.

To generate expressive diagrams (in analogy to [Mack]) means to map the relations and their properties onto graphical relations that match these properties. One graphical relation is the 'line' relation that connects arbitrary sorts of shapes (see figure 2, examples b) and c)). This graphical resource is very general and can thus be applied to any relation type. Another one, the 'inclusion' graphical relation constructs rectangles that include each other (see example d)). It can only be applied for tree order relations because it exactly matches the tree order properties. Relative position relations, such as 'object A sits below object B', can be applied to represent discrete linear orderings, for instance lists of elements. However, we use this graphical resource also in combination with the 'line' relation in which case it constrains the possible positions of the shapes in one dimension. The "line" relation that is constrained in such a way can visualize order relation type. Together with a legend 'attributes' can be used to represent bipartite relations (see examples a) and b)) and in the more specific case of a qualitative function its values can be visualised using the 'colour' resource. 'Length of box' can be used to represent quantitative function values such as time spans (see example). These examples show that we do the graphical binding at the relation types. Although this list of graphical relations is not complete it shows, by examples, how we assign graphical resources to the data. For a more detailed discussion of the assignment of graphical means of expression to the relation types we refer the reader to [Reichen].

The assigment of graphical relations to the types is not unique. Thus, in a given situation AVE has to decide what graphical relation, among those that are expressive for a certain data relation, is assigned to it. However, this can only be done in the context of the overall subnet to be visualized because the graphical means of expression are limited and because of that the different data relations have to compete with each other in a resource allocation process (see [Golo]). This process is constructive on the relation level. The variety of diagrams which is presented in figure 2 illustrates the different possible outcome of AVE's constructive design process. Thus, in example b) the 'continuous distance' graphical resource is combined with the 'line' graphical resource and this constructs the diagram. We do not have an explicit notion of a 'time–line' template or 'line' template that have to be combined in the process.

The aim of this paper is to discuss the final positioning algorithm by which the graphical elements are displayed given that a relation is visualised using the graphical relation 'line'. In order to do that we first sketch the interface between the overall visualisation system AVE and the spring–model algorithm. Then, we outline the constraints, that may be established during the resource allocation process. They affect the size, the position and the colour of the graphical objects. In this paper we only introduce the position constraints. After that we discuss the objective function which is succeded by a description of our gradient descent algorithm. Then we illustrate the results by presenting eight example layouts together with their computation times. Finally, we outline possible improvements.

The Interface to AVE

Assume we are given the connected graph $D = (V, E)$, where V is the set of nodes and E is the set of unordered edges $\{v_i, v_j\} \in E$ for $v_i, v_j \in V$. We further assume a set of constraints (as defined below) for the set of nodes. We let N be the total number of nodes to be represented in our graph, and in order to preserve the legibility of our layout, we

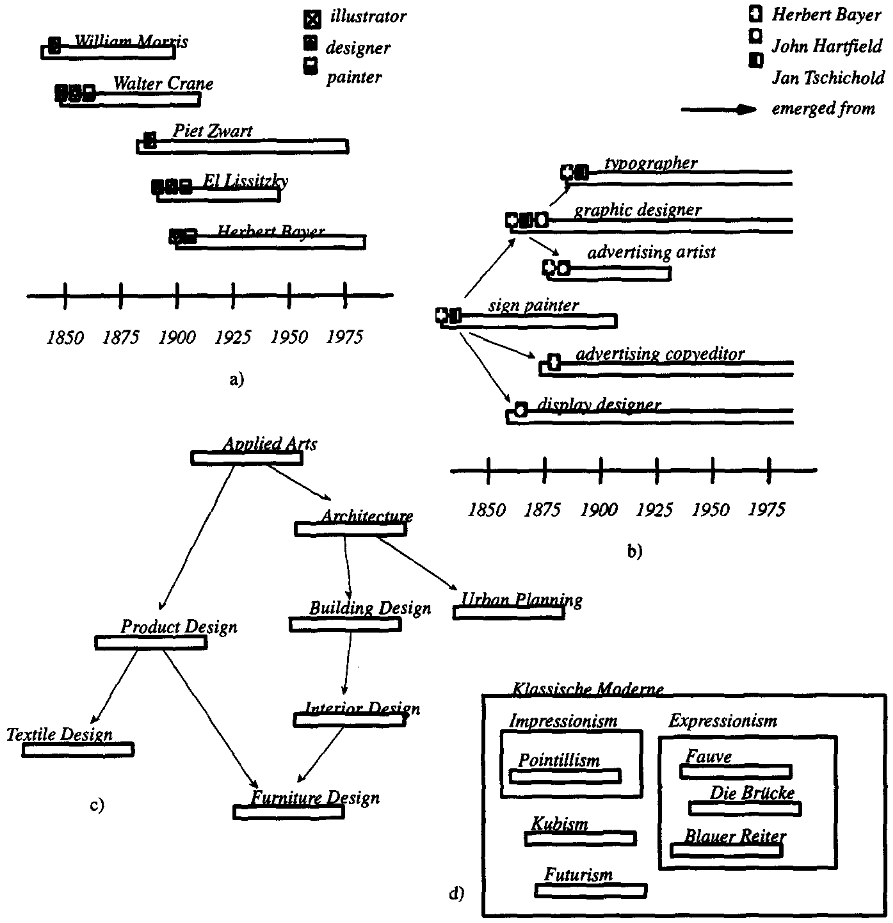


figure 2: some example diagrams

artificially assume N is less than or equal to 40 (neverthless, our algorithm may also handle graphs with a larger number of nodes). As already noted, each node (or object) is represented by a rectangle of fixed size (in contrast to taking them as points of a zero area). Thus, for each node we give three coordinates, “the left upper corner”, “the right lower corner” and the “center”. That is, $v_i = \{(x_i^L, y_i^T), (x_i^R, y_i^B), (x_i^C, y_i^C)\}$ where $x_i^L \le x_i^C \le x_i^R$ und $y_i^B \le y_i^C \le y_i^T$. Each rectangle contains a core area that represents a domain object the name of which is printed above this area. We take the center of the core area as the center of the overall rectangle. Therefore, the center of the ith rectangle does not correspond in general to the midpoint of the ith rectangle. Again, we allow for the size of each rectangle to differ from one another. We define $d(v_i, v_j) \in \mathrm{N}$ to be the topological distance between the two nodes v_i and v_j and this is simply the number of edges in the shortest path between the two nodes. Finally, we assume that the graph is to be represented in the square $\left[-\frac{1}{2}, \frac{1}{2}\right] \times \left[-\frac{1}{2}, \frac{1}{2}\right]$ with origin at $(0, 0)$.

The visualization machine AVE, in which our spring model algorithm is embedded, gives a set of geometrical constraints which we now list. As we have mentioned before, the constraints we consider here are means to express data relations geometrically. Thus, they are an important means for the communication of the data and their characteristics into graphics.

Fixed Positions Certain node positions can be fixed in either the x or y components. Thus, there is a (possibly empty) subset $P \sqsubseteq V$ such that for each $v_i = (x_i, y_i) \in P$ there exists p_i^x or $p_i^y \in \mathbb{R}$ with $x_i = p_i^x$ or $y_i = p_i^y$. The reader should note that by writing $v_i = (x_i, y_i)$ we are being intentionally vague as to what is being fixed, that is whether we are fixing x_i^L, x_i^C or x_i^C. This is certainly of little importance since we require that each individual object be of fixed size throughout the algorithm. Thus, $x_i^C - x_i^L$ and $x_i^R - x_i^C$ are both positive constants throughout.

Fixed Distances The distance between certain pairs of nodes can be fixed in either the x or the y direction. Thus, there is a (possibly empty) subset $D \subseteq V \times V$ such that for each $(v_i, v_j) \in D$ there exists d_{ij}^x or $d_{ij}^y \in \mathbb{R}$ with $x_i = x_j + d_{ij}^x$ or $y_i = y_j + d_{ij}^y$. Here $v_i = (x_i, y_i)$ and $v_j = (x_j, y_j)$.

Relative Positions A node can be restricted to lying above (below) to the right of (to the left of) another node. Thus, there is a (possibly empty) subset $R \subseteq V \times V$ such that for each $(v_i, v_j) \in R$ we have $x_i \leq x_j$ or $y_i \leq y_j$.

Orientation Two nodes can have a natural vertical or horizontal positioning relative to one another. Thus, there are (possibly empty) subsets $O_h \subseteq V \times V$ and $O_v \subseteq V \times V$ such that for each $(v_i, v_j) \in O_h$ (O_v) the orientation between v_i and v_j is horizontal (vertical). We will set $O = O_h \cup O_v$.

The Fixed Position and Fixed Distance constraints reduce the dimension of the problem from $2N$ to a smaller number depending on the magnitude of the sets P and D. The Relative Position constraint serves to restrict the domain of the objective function, which we will introduce below. The Orientation constraints are not absolute and will be incorporated in this objective function.

State of the Art in Spring Model Algorithms

[Eade] proposes a spring–model based algorithm which meets two aesthetic principles: firstly, all edges ought to have same length and secondly the layout should display as much symmetry as possible. He considers the graph that is laid out as a mechanical system in which the vertices of the graph represent steel rings and the edges represent spring forces. The input is a random initial placement of the vertices. His iteration process involves computing in each step the resultant force of the springs acting on each steel ring, which provide a force field whose action on the ring–spring system tends to move this system to a new position of lower potential energy. This is then repeated until a state of minimal energy is achieved.

The paradigm of the approach taken by Kamada [Kama] is the same as the one proposed by Eades. Here, the energy potential of the underlying mechanical system is given explicitly as an objective function

$$E = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{1}{2}(|p_i - p_j| - l_{ij})^2$$

that is to be minimised. In this formula $p_i = (p_i^x, p_i^y)$ and $p_j = (p_j^x, p_j^y)$ denote the positions of node i and node j and l_{ij} denotes the optimal distance between the two nodes. The aesthetic goal is that the Euclidean distance between any pair of nodes matches their topological distance. To realize this, Kamada implemented an optimization method which always finds a local minimum for E. An important difference to Eades' approach is that whenever a node is moved to another position all other nodes are frozen.

This is due to the 2n interdependent non–linear equations that they obtain from the minimisation condition (see [Kama]).

Both, Eades and Kamada, applied their algorithms to undirected graphs. The advantage of this technique is that it preserves symmetry without explicitly requiring knowledge of the automorphism group of the graph, a problem which is computationally expensive. Davidson and Harel [Davi] proposed a simulated annealing approach in which the energy potential of a system is described in terms of vertex distribution, nearness to borders, edge–lengths, and edge crossings. Fruchterman and Reingold [Fruc] presented an effective modification of Kamada's model. Sugiyama [Sugi] extended the spring model approach by introducing different sorts of magnetic forces that are applied in conjunction with the distance forces. One sensible way to apply magnetic forces is, e.g., to assign different edge types different orientations.

In contrast to the approaches we have discussed in this section our algorithm must consider a set of geometric constraints as introduced above. In our model, nodes are not just points in the plane, but rather are rectangles having varying x–and y–dimensions extents. Since our nodes are inhomogeneous, seeking a minimum distance between the centers of two nodes is not sufficient for the to minimization of overlapping. Therefore, in addition to 'distance forces', we introduce 'node–node repulsion forces' which serve to assure that the overlapping area of two nodes is minimized. Finally, we employ 'angle forces' in order to incorporate the Orientaion Constraints into the objective function.

In our first prototype [Klei] we extended Eades' approach by the additional forces introduced in the last paragraph. The algorithm we obtained this way was successful but rather slow. However, being a part of an interactive system, a major requirement any algorithm must meet is that diagrams should be laid out and displayed within interaction time. In order to satisfy this goal we now employ a gradient descent algorithm for the minimization of the objective function, which measures how well any given layout can be interpreted by the user.

The Objective Function

In the implementation of the objective function we take into account the "forces" arising due to the distance between pairs of nodes. Further, as we already mentioned we incorporate the "forces" arising from the predetermined orientation between pairs of nodes $(v_i, v_j) \in O$. The distance forces are calculated in such a way that this force is inversely proportional to the topological distance between pairs of nodes. We will adopt the following notation $\xi_{ij} = |x_i^C - x_j^C|$ and $\eta_{ij} = |y_i^C - y_j^C|$. Using this notation we can write the objective function for the 'distance forces' between the nodes v_i and v_j, similar to the formulation in [Kama], as

$$\rho_{ij}(v) = 1 - \frac{\sqrt{\xi_{ij}^2 + \eta_{ij}{}^2}}{d(v_i, v_j) \cdot a}$$

where alpha is the optimal distance between two nodes connected by a single edge. The optimal edge length would be, in the best of all possible cases, twice as long as the "standard" node. This length is approximated by the assumed average length of the string representing the object name. This is in contradistinction to [Kama] in which this optimal length is determined from the topology of the graph.

As we have indicated, the orientation constraints impose a natural horizontal or vertical positioning on any pair of nodes in the set O. The angle between the x–axis and the line through the nodes v_i and v_j is given by the formula

$$\gamma_{ij}(v) = \begin{cases} \tan^{-1}\left(\dfrac{y_j^C - y_i^C}{x_j^C - x_i^C}\right) \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right] & \text{if} \, x_i^C \neq x_j^C \\ 0 & \text{if} \, x_i^C = x_j^C \end{cases}$$

Then the deviation angles for any pair $(v_i, v_j) \in V \times V$ is

$$\theta_{ij}^h(v) = \begin{cases} \gamma_{ij} & \text{if } (v_i v_j) \in O_h \\ 0 & \textit{otherwise} \end{cases}$$

and

$$\theta_{ij}^v(v) = \begin{cases} \gamma_{ij} - \frac{\pi}{2} & \text{if } (v_i v_j) \in O_v \\ 0 & \textit{otherwise} \end{cases}$$

These are the objective functions which, upon differentiating, generate the 'angle forces' between the nodes v_i and v_j in the vertical direction and the horizontal direction, respectively. Now in as much as our nodes carry information for the user, we must insure that they do not overlap, thereby blocking part of that information from view. In order to achieve this, we consider the following functions

$$a_{ij}(v) = \begin{cases} \max\{x_i^R - x_j^L, 0\} & \text{if } x_i^L + x_i^R \leq x_j^L + x_j^R \\ \max\{x_j^R - x_i^L, 0\} & \text{if } x_j^L + x_j^R < x_i^L + x_i^R \end{cases}$$

$$\beta_{ij}(v) = \begin{cases} \max\{y_i^T - y_j^B, 0\} & \text{if } y_i^B + y_i^T \leq y_j^B + y_j^T \\ \max\{y_j^T - y_i^B, 0\} & \text{if } y_j^B + x_j^T < y_i^B + y_i^T \end{cases}$$

We take the product of a_{ij} and β_{ij} as the objective function for the 'node–node repulsion forces' between the nodes v_i and v_j and write $\psi_{ij}(v) = a_{ij}(v) \cdot \beta_{ij}(v)$

Upon a closer examination it is clear that ψ_{ij} is at least as large as the square of the area of the overlap of the i–th and j–th nodes. Further, if this pair of nodes do not overlap then $\psi_{ij} = 0$. With this notation we obtain the j–th term of our objective function, given by

$$f_j(v) = \sum_{i<j} \left(\frac{1}{d(v_i, v_j)^2} \rho_{ij}^2 + \omega_1 (\Theta_{ij}^{h\,2} + \Theta_{ij}^{v\,2}) + \omega_2 \psi_{ij} \right)$$

We have multiplied the square of the deviation angles by the scaling factor ω_1, which in actual calculations was always set equal to 0.05 and we have multiplied the overlap function ψ_{ij} by the factor ω_2. In order to weaken the forces between node pairs connected over topologically long paths, we have included the factor $\frac{1}{d(v_i, v_j)^2}$. For the sake of keeping the formula of the partial derivatives of $f_j(v)$ simple we first define the partial derivatives for its composites (we present only the derivatives in x) as

$$\frac{\delta \rho_{ij}(v)}{\delta x_i} = \begin{cases} \dfrac{-4\rho_{ij} \cdot \xi_{ij}}{d(v_i, v_j) \cdot \sqrt{\xi_{ij}^2 + \eta_{ij}^2}} & \text{if } x_i < x_j \\ 0 & \textit{otherwise} \end{cases}$$

$$\frac{\delta\Theta_{ij}^{v}(v)}{\delta x_i} = \begin{cases} 2\,\dfrac{\Theta_{ij}^{v}\,(y_j^C - y_i^C)}{(x_j^c - x_i^C)^2 + (y_j^c - y_i^C)^2} & \text{if } x_i > x_j \\ 0 & \textit{otherwise} \end{cases}$$

$$\frac{\delta\Theta_{ij}^{h}(v)}{\delta x_i} = \begin{cases} 2\,\dfrac{\Theta_{ij}^{h}\,(y_j^C - y_i^C)}{(x_j^c - x_i^C)^2 + (y_j^c - y_i^C)^2} & \text{if } x_i > y_j \\ 0 & \textit{otherwise} \end{cases}$$

$$\frac{\delta\psi_{ij}(v)}{\delta x_i} = \begin{cases} a_{ij} & \text{if } x_i^L + x_i^R \le x_j^L + x_j^R \text{ and } x_i^R \ge x_j^L \\ -\,a_{ij} & \text{if } x_j^L + x_j^R < x_i^L + x_i^R \text{ and } x_j^R > x_i^L \\ 0 & \textit{otherwise} \end{cases}$$

Thus, we can write the partial derivative of $f_j(v)$ in the following way

$$\frac{\delta f_j(v)}{\delta x_i} = \frac{\delta \rho_j(v)}{\delta x_i} + \frac{\delta\Theta_j^v(v)}{\delta x_i} + \frac{\delta\Theta_j^h(v)}{\delta x_i} + \frac{\delta\psi_j(v)}{\delta x_i}$$

$$\frac{\delta f_j(v)}{\delta y_i} = \frac{\delta \rho_j(v)}{\delta y_i} + \frac{\delta\Theta_j^v(v)}{\delta y_i} + \frac{\delta\Theta_j^h(v)}{\delta y_i} + \frac{\delta\psi_j(v)}{\delta y_i}$$

and by summing over j we get the objective function $f(v) = \sum_{j=1}^{N} f_j(v)$ whose gradient is constructed from the partial derivatives in the obvious way.

The Algorithm

Let us now consider an algorithm which will minimize the value of the objective function as defined in the previous section. We begin by reminding the reader that the gradient descent method can be applied successfully to any smooth function f over $\mathbb{R}^n$, which grow at infinity. Given an initial starting position $\vec{p}_i$, one calculates $\nabla f(\vec{p}_i)$ and defines τ_i as the smallest positive value which minimizes the function $\varphi(\tau) := f(\vec{p}_i - \tau\nabla f(\vec{p}_i))$. We then define $\vec{p}_{i+1} = \vec{p}_i - \tau_i \nabla f(\vec{p}_i)$. Iterating over $i \in \mathbb{N}$ gives us a sequence $(\vec{p}_i)$ which converges to a (local) minimum. We have modified this theoretically satisfying model in order to speed convergence and avoid problems one might encounter due to local minima. First we let

$m_{j,i} = \sqrt{\left(\dfrac{df(\vec{p}_i)}{dx_j}\right)^2 + \left(\dfrac{df(\vec{p}_i)}{dy_j}\right)^2}$ for $1 \le j \le N$ be the movement of node v_j in iteration i. Then, similar to the approach of [Fric], we define $\mu \in [0, \frac{\pi}{4}]$ to be the opening angle for oscillation detection and we let κ_i be the angle between the vectors given by the positions of node v_j during the i–2nd and i – 1st iteration. Given these definitions, we call

$$O_{j,i}(O_{j,i-1}, \kappa_i) = \begin{cases} O_{j,i-1} + r_0 * \cos(\kappa_i) & \text{if } |\cos(\kappa_i)| > \cos(\mu) \\ O_{j,i-1} & \textit{otherwise} \end{cases}$$

the oscillation factor of node v_j for $i > 2$ and $r_0 \in [0, 1]$. In contrast to their approach, we call the product $T_{j,i} = O_{j,i} * m_{j,i}$ of the oscillation factor with the movement of an arbitrary node v_j the temperature of this node. To give a formula for τ_i we need to define $T_{k,i} = \max\{T_{j,i} : 1 \le j \le N\}$ as the maximum temperature of the nodes in the ith iteration and from this we obtain the ratio $q_i = \frac{T_{k,i}}{T_{k,i-1}}$ that describes the maximum change of temperature between two consecutive iterations. Then we set

$$\tau_i = \begin{cases} \frac{c}{m_{k,i}} & \text{if } i \leq 2 \\ \tau_{i-1} * f & \textit{otherwise} \end{cases}$$

in which $c = \frac{a}{2}$. The factor

$$f = \begin{cases} 1 + w * (1 - q_i) & \text{if } q < 1,\ w \in [0, 0.5] \\ \frac{1}{q_i} & \textit{otherwise} \end{cases}$$

describes the correction of the τ_i with respect to the change of temperature. Such a choice of c and f, respectively, and hence τ_i have proven in practice to lead quickly to a very pleasing layout, and help control oscillations that can cause such an algorithm to become very inefficient. As a consequence, convergence is obtained if $T_{k,i} < \epsilon$ for some i which means that it depends only on the given graph and its characteristics and is not enforced by an artificial damping function nor by a limitation on the number of iterations. After the application of the above algorithm, it might be necessary to rescale the results so that the optimal layout is neither too large nor too small. We define the quantities

$$R_L = \min\{x_i^L | 1 \leq i \leq N\},\ R_R = \max\{x_i^L | 1 \leq i \leq N\},$$
$$R_T = \min\{y_i^T | 1 \leq i \leq N\},\ R_B = \max\{y_i^B | 1 \leq i \leq N\}.$$

that determine the bounding box of the graph.
Then we set $l_{\max} = \max\{|R_R - R_L|, |R_T - R_B|\}$, which we use as a scaling factor. Thus, we redefine the scaling factor $s_{i+1} = s_i \cdot \frac{1}{l_{\max}}$, for $s_0 = 1$ and translate the center of the bounding box to the origin in order to recenter the graph.

Results

In this section we will discuss example layouts that were generated by our spring–model algorithm. Along the first three diagrams we demonstrate how the semantics of the data may affect their visualization. Then, we present a couple of other example layouts some of which are taken from the literature. For each diagram we give the computation time which was measured on SPARCstation 20. The algorithm is implemented in ANSI–C. The three diagrams in figure 3 have the same topology if one abstracts from the directedness of the edges. Example a) represents a relation which is called 'work togteth-er'. It is defined on the set of 'artists' and it has the relation type Symmetric Relation. Therefore it is visualized using lines. Example b) represents the terminology relation 'broader term of' which is defined on the set of art concepts. It is qualified as an order relation on the class level but in the given instance it forms a tree–order relation. According to our model included rectangles would best visualize this graph. However, here we chose the 'arrow' resource in order to demonstrate the algorithm described in this paper. Thus, AVE decides to use the graphical relation 'below', which is defined on the y– coordinates, to constrain the positions of the nodes in the y–dimension. This way, we guarantee that the transitivity of the 'broader term of' relation is expressively visualized. In this case the Orientation constraints require the sons of a node to be vertivally positioned relative to the father node. Example c) represents an organization chart in which additionally the nodes on the same level have a fixed zero distance in the y coordinate.

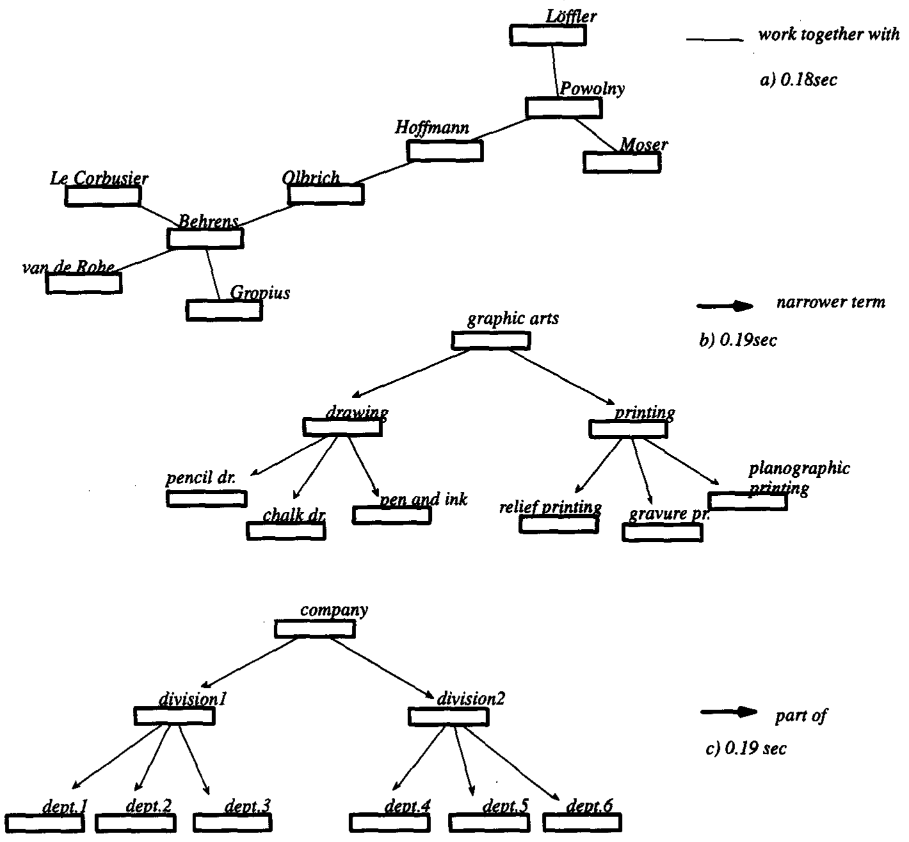
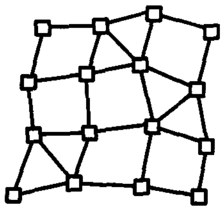


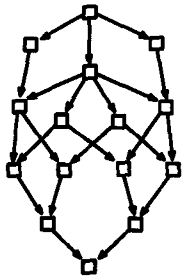
figure 3: Three graphs with the same topology but different relation types

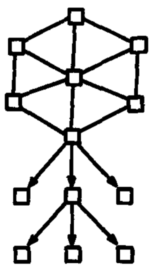
In figure 4 we present additional examples of layouts. Example a) and b) are unconstrained which implies that only the distance forces work which is in analogy to the algorithms of Eades and Kamada. The lattice in example c) is of order relation type visulized using arrows and 'below' constraints. Example d) the x–extents are determined by time intervals and thus we obtain fixed x–positions. In example e) we have defined 'below' constraints and fixed distance constraints for hierarchic subgraph whereas the remaining subgraph is not constrained at all.

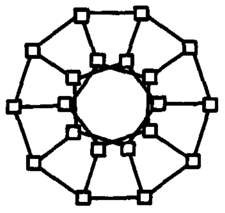
Summary and Future Work

In this paper we have presented a force directed placement routine which is an extension of the model originally proposed by [Eade]. Since we apply this routine in the context of the visualization algorithm AVE, we had to add two additional forces to the 'distance forces': the 'node–node repulsion force' that minimizes the overlapping area of two nodes and the 'angle force' that graphically realize a relative vertical (horizontal) positioning of two nodes to one another. Apart from the additional forces which we have modeled, a set of geometric constraints that are generated by AVE have to be integrated


a) example from [Eade]. 0.55 sec


c) example lattice, 0.50 sec


e) example from [Deng], 0.41 sec


b) example from [Kama], 0.89 sec

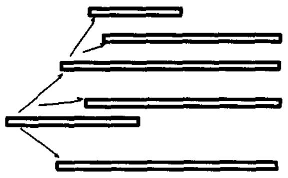

d) with arrows connected time–line example, 0.11 sec

figure 4: additional examples

into the graph layout process. Since a major requirement from the application is to insure interactivity the computation time was a crucial issue for us. This led us to the construction of an objective function with good differentiability properties in order exploit gradient descent methods and thus obtain a very fast algorithm producing expressive line diagrams effectively. However, we still encountered some problems which we consider should be solved in the future.

As can be seen in figure 4 example e), patterns such as the T–Shape template and the Hub–Shape template in the approach of [Kosa] may be visually realized just by using geometric constraints, that is, without explicitly having a notion of these patterns. However, this does not mean that the algorithm always preserves such patterns, for example, in the particular case when parts of the pattern are connected to other structures which themselves work against a construction of patterns. As mentioned before we consider the integration of such structural regularities as an important extension of our approach. However, this means to extend AVE's data analysis process in order to detect such regularities in the data and to assign them visualization templates.

Another problem occurs when the node extents are very inhomogeneous. In this case the ideal length of the edges between two nodes, which now relies on the size of the ideal node's extent, is no more a good measure. For this more general case we still have find a sensible notion of distance.

References

[Davi] Davidson R. Harel D., Drawing Graphs Nicely Using Simulated Annealing. Technical Report CS89–13, Department of Applied Mathematics and Computer Sciences, Weizman Institute Of Sciences, Israel, July, 1989.

[DiBa] Di Battista G. Eades P. Tamassia R. Tollis G., Algorithms for Drawing Graphs: an Annotated Bibliograpy, Brown University, Department of Computer Science, Technical Report, 1993.

[Deng] Dengler E. Friedell M. Marks J., Constraint–Driven Diagram Layout, Proceedings of the IEEE Symposium on Visual Languages, 1993.
[Eade] Eades P., A Heuristic Graph Drawing, Congressus Numerantium, 42, 1984.
[Fisch] Fischer D. H. Rostek L., SFK: A Smalltalk Frame Kit, Concepts and Use, To appear as GMD Report.
[Fric] Frick A. Ludwig A. Mehldau H., A fast Adaptive Layout Algorithm for Undirected Graphs (Extended Abstract and System Demonstration, in Graph Drawing, DIMACS International Workshop, GD '94, Princeton New Jersey, Ovtober 1994, Lecture Notes in Computer Science, Springer Verlag.
[Fruc] Fruchterman T. Reingold E., Graph Drawing by Force–Directed Placement. In Software Practice and Experience, 21(11), November 1991.
[Golo] Golovchinsky G. Kamps T. Reichenberger K, Subverting Structure: Data–driven Diagram Layout, accepted at IEEE Visualization '95, Atlanta, October 30–November 1 1995.
[Kamp94a] Kamps T. Reichenberger K. (1994a), Automatic Layout as an Organization Process, GMD Report, No. 825, Sankt Augustin 1994.
[Kamp94b] Kamps T. Reichenberger K. (1994b), Automatic Layout Based on Formal Semantics, in Catarci, T. et.al. (Eds.), Proceedings of the Workshop of Advanced Visual Interfaces, AVI '94, June 1194, Bari, Italy.
[Kama] Kamada T. Kawai S., An Algorithm for General Undirected Graphs, in Information Processing Letters 31 (1989). Elsevier Science Publishers B.V. (North Holland).
[Kosa] Kosak C. Marks J. Shieber S., Automating the Layout of Network Diagrams with Specified Visual Organization, in IEEE Transactions On Systems. MAn and Cybernetics, Vol 24, No. 3, March 1994.
[Klei] Kleinz J., Entwicklung eines constraint–gesteuerten 2D–Positionierungsverfahrens, to appear as GMD report.
[Lin] Lin T. Eades P., Integration of Declarative and Algorithmic Approaches for Layout Creation, in Proceedings of DIMACS International Workshop, GD '94 Princeton, New Jersey, USA, October 1994, Tamassia R. Tollis G. (Eds.), Lecture Notes in Computer Science 894,1994.
[Mack] Mackinlay J. (1986) Automating the Design of Graphical Presentations of Relational Information, in ACM Transactions on Graphics, 5(2).
[Reich] Reichenberger K. Kamps T. Golovchinsky G., Towards a Generative Theory of Diagram Design, accepted at IEEE Symposium on Information Visualization (Infovis '95), Atlanta, October 31 1995
[Roth] Roth S. F. Hefley W.E. Intelligent Multimedia Presentation Systems: Research and Principles, in Mark Marbury (Ed.) Intelligent Multi–Media Interfaces, AAAI Press, 1993.
[Sugi] Sugiyama K. Misue K. A Simple and Unified Method for Drawing Graphs: Magnetic–Spring Algorithm, in Proceedings of DIMACS International Workshop, GD '94 Princeton, New Jersey, USA, October 1994, Tamassia R. Tollis G. (Eds.), Lecture Notes in Computer Science 894,1994.