Drawing Directed Graphs Using One-Dimensional Optimization

Liran Carmel and Yehuda Koren

Dept. of Computer Science and Applied Mathematics
The Weizmann Institute of Science, Rehovot, Israel
{liran, yehuda}@wisdom.weizmann.ac.il

Abstract. We present an algorithm for drawing directed graphs, that works upon different principles than other algorithms in the field, producing the drawing by rapidly solving a sequence of three one-dimensional optimization problems. The algorithm can be directly applied to any kind of digraph, whether cyclic or not, without having to do any preprocessing such as inverting edge directions or introducing dummy nodes. We also derive a hierarchy index, which can quantitatively measure the amount of hierarchy, or directionality, in a given digraph.

1 Introduction

Visualizing directed graphs (digraphs) is a challenging task, requiring to faithfully represent the relative similarities of the nodes, as well as to give some sense to the overall directionality. This last requirement makes the many algorithms designed for undirected graphs inappropriate for use with digraphs. Consequently, algorithms for digraph drawing usually adopt different strategies from their undirected counterparts. The dominant strategy, rooted in the work of Sugiyama *et. al.* [16], dictates axes separation — separate determination of the y, and then the x coordinates of the nodes, such that the y-axis is devoted to represent the directional information, or hierarchy, and the x-axis allows for additional aesthetic considerations, such as shortening edge lengths or minimizing the number of edge crossings.

Assigning the y-coordinates is done by dividing the y-axis into a finite number of layers, and associating each node to exactly one layer — a process which is appropriately called *layering*. No edge is allowed between nodes residing in the same layer. Actually, edges are usually permitted only between nodes from neighboring layers, and whenever an edge should cross a layer, a *dummy node* is inserted to prevent this. The many existing variants of layering algorithms are all motivated by the simple observation that a layering in which all edges point to the same direction exists if and only if the digraph is acyclic. This, of course, raises the question what to do when dealing with cyclic digraphs. In this case the custom is to apply a preliminary stage that turns the digraph to acyclic by inverting the direction of a minimal number of edges. An optimal solution of this task is NP-hard, but several sub-optimal algorithms have been proposed, see details in, e.g., [3, 12].

Assigning the x-coordinates is normally done in two stages. The first determines the order of the nodes within each layer, in an iterative process called *ordering*. In a single iteration, the order of the nodes in all layers but one is fixed, and the order of the mobile nodes is determined such as to minimize the number of edge crossings. This is, too, an NP-hard problem, with which one should confront. The second stage determines the exact locations of the nodes along the x-axis, taking into account various parameters, such as the finite size of the nodes and the smoothness of the edges.

Digraph drawing algorithms have been evolved to produce very nice layouts for many different types of digraphs. Yet, we would like to point out two inherent properties of the standard strategy that, despite being treated in various ways by the many algorithms, might in some cases be undesirable:

- Finding a layering for cyclic digraphs necessitates their transformation into acyclic ones, thus introducing an unavoidable distortion of the original problem. Another distortion, although less significant, is caused by the insertion of the dummy nodes.
- The layering is strict in the sense that the y-axis is quantized into a finite number of layers. This constraint may sometimes be advisable for acyclic digraphs, but we show that allowing for more flexibility turns out to be advantageous to the drawing.

In this paper we present a new algorithm for digraph drawing. Our algorithm embraces the idea of axes separation, but uses novel approaches for drawing both axes. Theses approaches, besides from providing nice drawings and having fast implementations, also successfully deal with the two aforementioned points of the distortion and the discrete layering.

We associate with the nodes continuous y-coordinates in a way that suggests a natural unified framework which can be applied to any kind of digraph, whether cyclic or acyclic, requiring no modifications what so ever to the graph. In particular, dummy nodes are not required, and cyclic digraphs do not have to go through the process of edge inversion. For some digraphs, the continuous layering produces the usual quantization of the y-axis. But, for many other digraphs the quantization is broken, to better represent the hierarchy.

We define the vector of y-coordinates as the unique minimizer of a simple energy function. We show that the minimization problem is equivalent to a system of linear equations, whose solution can be found with high speed. The simple form of the energy function enables its rigorous analysis, giving rise to many interesting results, maybe the important of which is the definition of an index to measure the amount of hierarchy in a digraph.

In the absence of strict layering we cannot use traditional schemes for drawing the x-coordinates, since now the stage of ordering becomes meaningless. It would be natural to adopt force directed models, like those commonly used for drawing undirected graphs. We have developed a model suited for our one-dimensional problem. Our optimization process takes place in two parts: Instead, we use a different two-stage approach: first we find a crude vector of coordinates by minimizing an energy function, constructed upon aesthetical considerations. Then, we beautify the result by applying a one-dimensional force directed model that assumes attractive forces, as well as repulsive ones, between the nodes.

By definition, a force is minus the gradient of the energy. Thus, the strategy of energy minimization is equivalent to a force directed model. Therefore, had we been asked to categorize our algorithm, we would have say that it is purely energy minimization oriented. All of its parts are using energy minimization procedures, each part with its own especially tailored energy function. Force directed models are much more popular in undirected graph drawing than it is in digraph drawing. Probably, the majority of the undirected graph drawing algorithms are of this type, would it be by directly assigning forces between the nodes [5, 6], or by minimizing energy functions [11, 2].

We are aware of only one other occasion where a force directed model was suggested for the benefit of digraph drawing [15], forcing directionality by applying a homogeneous magnetic field and favoring edges that are parallel to its field lines. Yet, we are under the impression that the inferred energy function is complicated, rich in local minima, and difficult to minimize.

2 The Algorithm

A digraph is usually written as $G(V, E)$, where $V = \{1, \ldots, n\}$ is a set of n nodes, and $E \subseteq V \times V$ is a set of directed edges, (i, j) being the edge pointing from node i to node j. Hereinafter we reserve the notation (i, j) to describe a directed edge, and use $\langle i, j \rangle$ to simply state that there is an edge between nodes i and j. Our drawing algorithm enables the enrichment of this definition, making it more quantitative by adding two ingredients:

1. **Edge weights**: Let w_{ij} be the weight associated with the edge connecting nodes i and j. We assume that the weights satisfy three properties:
   - $w_{ij} \geq 0 \quad \forall i, j$.
   - $w_{ii} = 0 \quad \forall i$ (no self-edges).
   - $w_{ij} = 0 \quad$ for i, j non-adjacent pair.

   We symbolize by W the $n \times n$ matrix of weights. Obviously, this is a symmetric matrix, $w_{ij} = w_{ji}$.
2. **Target height differences**: Let $\langle i, j \rangle$ be an adjacent pair. We express the relative hierarchy of nodes i and j by the number δ_{ij}, measuring their desired height difference along the y-axis. This is, in the drawing we would like to place nodes i and j such that $y_i - y_j = \delta_{ij}$. We symbolize by Δ the $n \times n$ matrix of target height differences. By definition, this is an antisymmetric matrix, $\delta_{ij} = -\delta_{ji}$.

Correspondingly, from this point on we write a digraph as $G(V, W, \Delta)$. This wider definition does not reduce generality, since, in the absence of information on the weights and/or the target height differences, it is always possible to associate with each edge (i, j) the default values $w_{ij} = \delta_{ij} = 1$, and to set $w_{ij} = 0$ for any non-adjacent pair. Hereinafter, we shall call a digraph with these default values by the name *unweighted digraph*, and denote its weights matrix and target height differences matrix as W^0 and Δ^0, respectively. The only ingredient incorporating directional information is Δ. A digraph with $\Delta = 0$ (i.e., $\delta_{ij} = 0$ for any i and j) is nothing but an undirected graph.

For later use we define two magnitudes associated with a digraph $G(V, W, \Delta)$ — the *Laplacian* and the *balance*:

Definition 1 (Laplacian). *Let $G(V, W, \Delta)$ be a digraph. The Laplacian of the digraph is the symmetric $n \times n$ matrix*

$$L_{ij} = \begin{cases} \sum_{k=1}^{n} w_{ik} & i = j \\ -w_{ij} & i \neq j \end{cases} \qquad i, j = 1, \ldots, n.$$

This is just the conventional definition of Laplacian, customary for undirected graphs. Indeed, since the Laplacian is independent of Δ, the definition does not distinguish digraphs to undirected graphs. The Laplacian has a key role in some undirected graph drawing algorithms, see, e.g., a previous work of ours [13], and will be shown to have fundamental part here, too. One of its most important properties is the following:

Lemma 1. *Let $G(V, W, \Delta)$ be a digraph. Its Laplacian is a positive semi-definite matrix, thus having non-negative real eigenvalues. Moreover, when G is a connected digraph, L has exactly one zero eigenvalue, corresponding to the eigenvector $c \cdot 1_n$, where $1_n = (1, \ldots, 1)^T$ and c any constant.*

Proof. See Hall [8], or Koren *et al.* [13]. □

Naturally, a drawing algorithm has to deal only with connected digraphs. When a digraph is disconnected, one should separately draw each of its connected sub-digraphs. Unless otherwise stated, we shall hereinafter always assume a connected digraph.

Definition 2 (Balance). *Let $G(V, W, \Delta)$ be a digraph. The balance of the i'th node, denoted b_i, is*

$$b_i = \sum_{j=1}^{n} w_{ij}\delta_{ij}.$$

The balance of G is the vector

$$b = (b_1, \ldots, b_n)^T.$$

A node whose balance is zero will be called a balanced node.

The balance of the i'th node measures the difference between how much it pushes away other nodes (those nodes j for which $\delta_{ij} > 0$), and how much is it pushed away (by those nodes j for which $\delta_{ij} < 0$). For the unweighted digraph the balance is simply the difference between the out-degree and the in-degree of the node, thus the name balance. The balance vector has the following useful property:

Lemma 2. *Any balance vector b is orthogonal to the vector 1_n,*

$$b \cdot 1_n = \sum_{i=1}^{n} b_i = 0.$$

Proof. From Definition 2

$$\sum_{i=1}^{n} b_i = \sum_{i,j=1}^{n} w_{ij}\delta_{ij} = 0,$$

where the last equality follows from the fact that W is symmetric while Δ is antisymmetric. □

In the remaining of this section we develop the theory that underlies the different parts of the algorithm. Details of implementation are postponed to section 3.

2.1 Assigning the y-Coordinates

We suggest to use an energy function, whose minimization yields a vector of coordinates that bears some desired properties.

Definition 3 (Hierarchy Energy). *Let $G(V, W, \Delta)$ be a digraph, and let $y = (y_1, \ldots, y_n)^T$ be any vector of coordinates. The hierarchy energy is*

$$E_H(G, y) = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(y_i - y_j - \delta_{ij})^2 = \sum_{(i,j)\in E} w_{ij}(y_i - y_j - \delta_{ij})^2. \qquad (1)$$

Clearly, $E_H \geq 0$ for any digraph and any vector of coordinates. We define an *optimal arrangement* of a digraph, $y^\star$, as a minimizer of the hierarchy energy, $y^\star = \arg\min_y E_H(G, y)$. An optimal arrangement will try to place the nodes such that the height difference $y_i - y_j$ for any adjacent pair $\langle i, j\rangle$ will be close to δ_{ij}. The weight w_{ij} indicates how "important" it is that $y_i - y_j - \delta_{ij}$ would be small. The larger it is, the smaller should be $(y_i - y_j - \delta_{ij})^2$ to keep the contribution to the energy small.

The hierarchy energy can be written in a more compact form, using the previously defined notions of Laplacian and balance:

Lemma 3. *Let $G(V, W, \Delta)$ be a digraph with Laplacian L and balance b, and let $y = (y_1, \ldots, y_n)^T$ be any vector of coordinates. The hierarchy energy is given by*

$$E_H = E_0 + y^T L y - 2y^T b, \tag{2}$$

where $E_0 = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}\delta_{ij}^2$.

Proof. Expanding the hierarchy energy (1) we get

$$E_H = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(y_i - y_j)^2 - \sum_{i,j=1}^{n} w_{ij}\delta_{ij}(y_i - y_j) + \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}\delta_{ij}^2.$$

The first term was shown in [8] to be just $y^T L y$. The third term is, by definition, E_0. The second term is

$$-\sum_{i,j=1}^{n} w_{ij}\delta_{ij}(y_i - y_j) = -2\sum_{i,j=1}^{n} w_{ij}\delta_{ij}y_i = -2y^T b,$$

where the first equality stems from W being symmetric and Δ being antisymmetric. □

Exploiting the simple form of the hierarchy energy, we find an explicit formula for an optimal arrangement. As the next result shows, $y^\star$ is the solution of a system of linear equations.

Proposition 1. *Let $G(V, W, \Delta)$ be a digraph, with Laplacian L and balance b. An optimal arrangement $y^\star$ is a solution of*

$$Ly^\star = b.$$

Proof. Differentiating (2) with respect to y and equating to zero gives:

$$\frac{\partial E_H}{\partial y} = 2Ly - 2b = 0.$$

Thus, the solution of $Ly = b$ corresponds to an extremum of E_H. It is a global minimum, since E_H has a quadratic form with $y^T L y \geq 0$ (recall from Lemma 1 that L is positive semi-definite). □

But what can we say about existence and uniqueness of this solution? Lemma 1 tells us that L is singular. However, this should not worry us, as the following proposition shows.

Proposition 2. *Let $G(V, W, \Delta)$ be a connected digraph, with Laplacian L and balance b. The system $Ly = b$ is compatible, with an infinite number of solutions differing only by a translation.*

Proof. An existence of at least one solution follows from the fact that the energy E_H is bounded from below.
Suppose that y^1 and y^2 are two solutions of $Ly = b$, i.e., $Ly^1 = b$ and $Ly^2 = b$. Therefore, $L(y^2 - y^1) = 0$, with $y^2 - y^1$ an eigenvector of L corresponding to the zero eigenvalue. From Lemma 1 it must be that $y^2 - y^1 = c \cdot 1_n$. □

The uniqueness (up to a translation) suggests that $y^\star$ carries some essential information. Indeed, as will be shown in section 4, this exact property is the one that makes feasible the definition of an hierarchy index.

Actually, proposition 2 enables us to define the optimal arrangement in a completely unique fashion. We require that the center of mass of the optimal arrangement is in the origin of coordinates, i.e., $\sum_i y_i^\star = 0$. Therefore, we redefine the optimal arrangement as:

Definition 4 (Optimal Arrangement). *Let $G(V, W, \Delta)$ be a digraph with Laplacian L and balance b. Its optimal arrangement, $y^\star$, is the solution of $Ly^\star = b$, subject to the constraint $(y^\star)^T \cdot 1_n = 0$.*

This choice of $y^\star$ enables its fast computation, see section 3.1. A delicate situation occurs when b is the zero vector, since then the optimal arrangement is in itself the zero vector. The y-axis, then, contributes nothing to the drawing, assigning the same coordinate to all nodes. This topic will be covered in section 4.

Let us give an impression on how this algorithm works by applying it to a few small scale examples. More examples will be brought in later sections. The digraph in Figure 1(a) is an unweighted acyclic one. Its optimal arrangement is the solution of the system

$$\begin{pmatrix} 2 & -1 & -1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} y^\star = \begin{pmatrix} 2 \\ -1 \\ -1 \end{pmatrix},$$

under the constraint $(y^\star)^T \cdot 1_n = 0$. This gives

$$y^\star = \begin{pmatrix} 2/3 \\ -1/3 \\ -1/3 \end{pmatrix},$$

which is just the expected two-layer solution. The height difference between the layers is 1, thus $\delta_{ij} = y_i - y_j$ for all $\langle i, j \rangle$, giving $E_H(y^\star) = 0$.

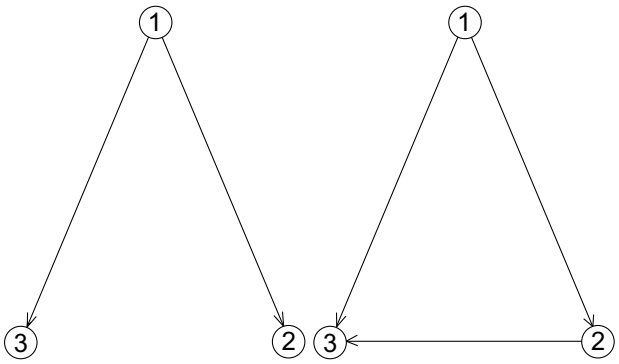


Fig. 1. Two examples of unweighted acyclic digraphs, for which we find in the text their optimal arrangement.

The digraph in Figure 1(b) is another example of an unweighted acyclic digraph. Now the optimal arrangement is

$$y^\star = \begin{pmatrix} 2/3 \\ 0 \\ -2/3 \end{pmatrix},$$

which is the constrained solution of the system

$$\begin{pmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix} y^\star = \begin{pmatrix} 2 \\ 0 \\ -2 \end{pmatrix}.$$

Aesthetically, this vector of coordinates impressively captures the structure of the digraph. In contrast to the previous example, nodes 2 and 3 can no longer have the same y-coordinate since they push each other in opposite directions. The result reflects a sort of a compromise, pushing node 2 upwards, thus decreasing the height difference $y_1 - y_2$ to $\frac{2}{3}$, while pushing node 3 downwards, thus increasing the height difference $y_1 - y_3$ to $\frac{4}{3}$. The height differences cannot achieve their targets, resulting in a strictly positive hierarchy energy $E_H(y^\star) = (\frac{2}{3} - 1)^2 + (\frac{2}{3} - 1)^2 + (\frac{4}{3} - 1)^2 = \frac{1}{3}$.

The digraph in Figure 2(a) is an example of an unweighted cyclic digraph. This time the system to be solved is

$$\begin{pmatrix} 2 & -1 & 0 & -1 \\ -1 & 3 & -1 & -1 \\ 0 & -1 & 2 & -1 \\ -1 & -1 & -1 & 3 \end{pmatrix} y^\star = \begin{pmatrix} 0 \\ 1 \\ 0 \\ -1 \end{pmatrix},$$

giving

$$y^\star = \begin{pmatrix} 0 \\ 1/4 \\ 0 \\ -1/4 \end{pmatrix},$$

which is schematically plotted in Figure 2(b). Here we see the ease and naturalness by which our algorithm deals with cyclic digraphs. The result is aesthetically convincing, putting node 2, whose balance is the largest, at the top, and node 4, whose balance is the smallest, at the bottom. As is always the case with cyclic digraphs, the height differences cannot all achieve their targets, reflecting in strictly positive hierarchy energy. Indeed, $E_H(y^\star) = 4 \cdot (\frac{1}{4} - 1)^2 + (\frac{1}{2} - 1)^2 = 2.5$.

The idea of using energy minimization to determine a vector of coordinates per one axis of the drawing, was already exploited in the field of undirected graph drawing. Two of the most successful realizations of this idea were introduced by Tutte [17] and Hall [8], both utilizing the same quadratic energy function. We next show that the hierarchy energy can be viewed as a generalization of the Tutte-Hall energy, suggesting the possibility of drawing undirected graphs and digraphs using the same tools.

Tutte and Hall used the following quadratic energy function:

$$E_{TH} = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(y_i - y_j)^2 = y^T L y.$$

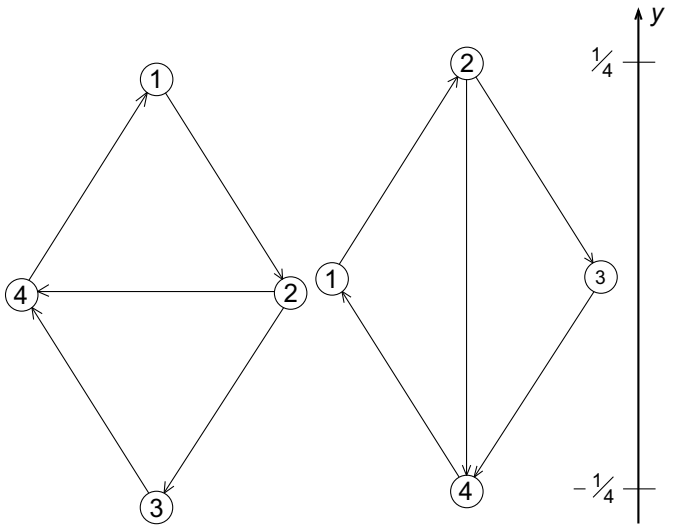


Fig. 2. (a) An example of an unweighted cyclic digraph. **(b)** Its optimal arrangement.

Comparing this energy with the hierarchy energy (1) or (2), it is clear that they identify for digraphs with $\Delta = 0$, which are nothing but undirected graphs. Farther than that, undirected graphs are members of the larger family of zero balance vector digraph, for which we get the (undesirable) zero vector as a minimizer, $y^\star = (0, \ldots, 0)^T$.

The case of a zero balance vector is discussed in section 4, and here we just briefly explain how did Tutte and Hall deal with it in the framework of undirected graph drawing:

- **Tutte's solution**: Tutte arbitrarily chose a certain number of nodes to be anchors, i.e., he fixed their coordinates in advance. This, of course, prevents the collapse formerly imposed by (3), but instead raises new problems, such as what should be the number of anchors, how to determine their coordinates, and why after all such an anchoring mechanism should give nice drawings.
- **Hall's solution**: Hall constrained the solution to be centered at the origin, which is just a translation. Furthermore, Hall posed an overall scaling constraint $y^T y = 1$, thus avoiding the zero-vector solution. He showed that the optimal vector of coordinates is the eigenvector of the Laplacian that corresponds to the lowest positive eigenvalue. This vector, also known as the Fiedler vector, is of tremendous importance in many other fields too. This approach, so it seems, yields nice drawings, see examples in [8] and [13].

It is instructive to adopt a different viewpoint in explaining a fundamental difference between the minimizer of the Tutte-Hall energy, and the optimal arrangement $y^\star$. The former is obtained from the equation $\partial E_{TH}/\partial y_i = 0$ which gives

$$y_i = \frac{\sum_{j=1}^{n} w_{ij} y_j}{\sum_{j=1}^{n} w_{ij}}. \tag{3}$$

This equation tells us to put node i in the *barycenter* of its neighbors. Clearly, the zero vector is a solution of (3), a situation that both Tutte and Hall avoid by using various

constraints. In analogy, the minimizer of our hierarchy energy is obtained from

$$\frac{\partial E_H}{\partial y_i^\star} = 2\sum_{j=1}^{n} w_{ij}(y_i^\star - y_j^\star - \delta_{ij}) = 0.$$

This gives the following important property of $y^\star$:

$$y_i^\star = \frac{\sum_{j=1}^{n} w_{ij}(y_j^\star + \delta_{ij})}{\sum_{j=1}^{n} w_{ij}},$$

which is substantially different from (3). Here we take a *"balanced" weighted average* instead of a barycenter. The introduction of non zero δ_{ij}'s prevents the collapse of all the nodes to the same location, yielding a meaningful solution.

2.2 Assigning the x-Coordinates

In principle, we would like to use a classical force directed model for the x-axis. Directional information should not be considered any longer, since it is assumed to be exhaustively taken care of by the y-axis. However, when trying to modify the customary two-dimensional gradient descent optimization algorithm, for use in our one-dimensional case, convergence was rarely achieved. The reason for this is what we call the swapping problem. One should remember that the y-coordinates of the nodes were already fixed by the layering, and now the nodes are allowed to move only along the x-axis. But, if two nodes have close y-coordinates, swapping places along the x-axis is almost always impossible, even if it is energetically favorable. This is demonstrated in Figure 3. Suppose we have nodes 1 and 2, whose y-coordinates are close, arranged along the x-axis as shown in Figure 3(a). Suppose also that the arrangement in Figure 3(b) is energetically favorable. Yet, the transition from state (a) to state (b) involves an intermediate state like the one shown in Figure 3(c), in which the nodes become very close. In this state the repulsive forces are the dominant, preventing farther progress of the nodes.

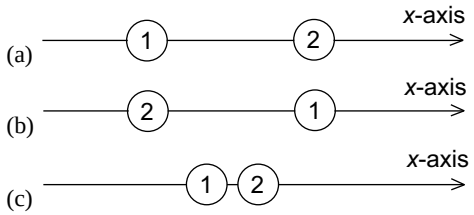


Fig. 3. Visualization of the swapping problem.

It would be the best, then, to use alternative optimization technique for our one-dimensional case, which skirts the swapping problem. We achieve this by using a two-stage scheme: first, we find the vector of coordinates by using a simple energy function that can be minimized by powerful global techniques. This energy function does not depend on the already known y-coordinates. Then, we use a more complex, and rich, y-dependent energy function, to achieve local beautification of the result.

Using this scheme made our one-dimensional problem substantially faster and easier to solve than two-dimensional force directed models. After all, this is what one would expect considering the significantly lower number of topological configurations in one dimension.

The First Stage Here, we forget about the y-coordinates of the nodes. Our goal is to find a vector $x = (x_1, \ldots, x_n)^T$, representing the x-coordinates of the nodes, in a way that fulfills some global aesthetic properties.

We present two variants for the algorithm. One finds a vector of coordinates by minimizing edge lengths, and the other does it by minimizing edge squared lengths. Either way, the aesthetical reasoning is clear. Moreover, minimizing edge lengths, or their squares, is known to reduce the number of edge crossings in the final drawing.

– **Minimizing edge squared lengths**: This means minimizing the already familiar Tutte-Hall energy function,

$$E_{TH} = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(x_i - x_j)^2 = x^T L x.$$

As discussed in section 2.1, the non-trivial minimizer of this energy function is the Fiedler vector which is the eigenvector of the Laplacian associated with the smallest positive eigenvalue. We find the minimizer of E_{TH} using ACE [13], which is an extremely fast multiscale algorithm for undirected graph drawing which computes, among other things, the Fiedler vector.

– **Minimizing edge lengths**: This is the well known problem of *minimum linear arrangement* [4]. The solution is obtained by minimizing the energy function

$$E_{LA} = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}|x_i - x_j|,$$

where $(x_1, \ldots, x_n)$ is a permutation of $(1, \ldots, n)$. We find the minimizer of E_{LA} using another fast multiscale algorithm [14], designed especially for this problem.

In most of the cases we have studied, the Fiedler vector was inferior with respect to the final result. The reason for this is that nothing in the Tutte-Hall energy function prevents from two nodes to have the same x-coordinate. Therefore locally dense regions are probable to appear. In contrast, the linear arrangement energy incorporates some kind of a "repulsive force", by not letting two nodes having the same x-coordinate.

The Second Stage In the first stage we have found the vector x by using a very simple energy function, and ignoring the information about the y-coordinates of the nodes. As a result, there are cases where a local smoothing can improve the drawing. We achieve this using a richer one-dimensional force directed model that incorporates the information about the y-coordinates.

We have been choosing to design a one-dimensional version of the Fruchterman-Reingold model [6], but we would like to emphasize that this is not the only possible option. Other force models that could have been used in probably equal success are, to name but a few, the simulated annealing energy model [2] that can deal with edge corssings, or the force directed model of Eades [5].

The original two-dimensional model of Fruchterman and Reingold assumes attractive forces, as well as repulsive ones, between the nodes. The attractive forces care that strongly connected nodes will have close x-coordinates, and the repulsive forces restrain the attraction by preventing nodes from becoming too close. Every adjacent pair $\langle i, j \rangle$ is subject to an attractive force of magnitude d_{ij}^2/K. Here, $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ is the distance between the nodes, and K a positive constant. Also, any two nodes i and j, not necessarily adjacent, are subject to a repulsive force of magnitude K^2/d_{ij}, with K the same constant as before. The bigger is K, the more dominant would the repulsive force be, and the more "open" would be the drawing. The two forces work along the straight line connecting the nodes. Clearly, this stage does utilize the already known y-coordinates. We keep this exact same form of the forces, but let the node move only along the x-axis.

3 The Implementation

Here we discuss implementation details of the different parts of the drawing algorithm.

3.1 Assigning the y-Coordinates

Finding the optimal layering means solving the system $Ly^\star = b$, see Definition 4. Classical solvers, however, might fail since, by Lemma 1, L is singular. However, here comes to an effect the fundamental importance of the fact that we have been defining the optimal arrangement to be orthogonal to 1_n. As will be shortly explained, this fact enables us to solve the system $Ly^\star = b$ by direct application of a conjugate gradient algorithm [7], see Figure 4. All that we have to do is to make sure that the initial vector would be orthogonal to 1_n. A convergence is guarantied in n iterations, but in practice is much faster than that. Each iteration is fast, with the dominant operation being a single multiplication of a matrix with a vector, which is done in time $O(|E|)$, where E is the set of edges.

For the interested reader we now show why we can safely use a conjugate gradient algorithm on the system $Ly^\star = b$. The basic idea is that the singularity of L can be removed by restricting the problem to a subspace of $\mathbb{R}^n$.

Let P be the rotation matrix that sets the direction 1_n to be the first axis of $\mathbb{R}^n$, i.e.,

$$P \cdot 1_n = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Let $b' = Pb$ be the rotated balance, and $y^{\star'} = Py^\star$ the rotated optimal arrangement. Then $L'y^{\star'} = b'$, where $L' = PLP^T$ is the rotated Laplacian.

¿From Definition 4, $y^\star \cdot 1_n = 0$, so that $y_1^{\star'} = 0$. Also, from Lemma 2 $b \cdot 1_n = 0$, thus $b'_1 = 0$. Therefore, the first element of both $y^{\star'}$ and b' is identically zero, suggesting the possibility of ignoring that axis and restricting the problem to the orthogonal $(n-1)$-dimensional subspace. Let b'' be the $(n-1)$-dimensional vector $(b'_2, \ldots, b'_n)^T$, and let $y^{\star''}$ be the $(n-1)$-dimensional vector $(y_2^{\star'}, \ldots, y_n^{\star'})^T$. Let L'' be the $(n-1) \times (n-1)$ matrix obtained by omitting the first row and first column of L'. Then

$$L''y^{\star''} = b'',$$

Function Conjugate_Gradient (L, b)
% L — the Laplacian
% b — the balance

% Initialization:
$y_{(0)} \leftarrow$ choose at random
$y_{(0)} \leftarrow (y_{(0)} \cdot 1_n) \cdot \frac{1_n}{\|1_n\|}$ % Orthogonalization
$d_{(0)} \leftarrow r_{(0)} \leftarrow b - Ly_{(0)}$
$i \leftarrow 0$

while $\|r_i\| >$ tolerance
$\alpha_{(i)} \leftarrow \frac{r_{(i)}^T r_{(i)}}{d_{(i)}^T L d_{(i)}}$
$y_{i+1} \leftarrow y_{(i)} + \alpha_{(i)} d_{(i)}$
$r_{i+1} \leftarrow r_{(i)} - \alpha_{(i)} L d_{(i)}$
$\beta_{i+1} \leftarrow \frac{r_{(i+1)}^T r_{(i+1)}}{r_{(i)}^T r_{(i)}}$
$d_{i+1} \leftarrow r_{(i+1)} + \beta_{(i+1)} d_{(i)}$
$i \leftarrow i + 1$
end while
return $y_{(i+1)}$

Fig. 4. The conjugate gradient algorithm that we use to find the optimal arrangement.

where it can be easily proved that L'' a positive definite, thus non-singular, matrix.

A system of linear equations with a positive definite matrix is known for being solved in high speed and with excellent numerical stability. If L'' is dense, one can solve the system using Cholesky factorization [7], which is faster and more stable than Gauss elimination. If L'' is sparse, iterative techniques should be preferred. Gauss-Seidel relaxation [7] is guarantied to converge, but the conjugate-gradient (CG) technique [7] is considered stronger, with a guarantied convergence in n iterations. L'' is used in each iteration by multiplying it once with a vector. In fact, this multiplication is the only way by which L'' is used. For our case, this fact is of utmost practical importance, sparing the need to go through the process of restricting L to L''. Any multiplication of the form $L \cdot y$ where y is orthogonal to 1_n, keeps us in the subspace orthogonal to the direction 1_n. Thus, as long as we start with a vector orthogonal to 1_n, we are guarantied that the final solution $y^\star$ will be orthogonal to 1_n, too. This justify why we can use the conjugate gradient algorithm directly on $Ly^\star = b$.

3.2 Assigning the x-Coordinates

The implementation of the first stage involves familiar methods that were already developed for other purposes, [13, 14], see section 2.2. Here, we describe in more details the implementation of the second stage.

In the original two-dimensional model of Fruchterman and Reingold we iteratively change the location of the nodes in the plane. The direction of a single move is determined by a gradient descent algorithm, and its magnitude, usually known as the *temperature*, is a decreasing function of time.

In the one-dimensional analog, the direction loses most of its sense. The infinite number of possible directions in a plane, degenerates into only two, namely right and left. Consequently, the only interesting output of a one-dimensional gradient descent algorithm, is the sign of the force. As is explained in the next section, we do not even use this information. In fact, we do not apply a gradient descent algorithm at all, and use the Fruchterman-Reingold forces only via their induced energy function. The original Fructherman-Reingold model defines only forces, and we have derived the energy induced by them using the relation $E = -\int f(x)dx$, where E stands for energy and f stands for force. This gives the following expression for what we might call the *Fruchterman-Reingold energy function* (constant terms have been omitted):

$$E_{FR} = \frac{1}{3K} \sum_{\langle i,j \rangle \in E} d_{ij}^3 - K^2 \sum_{i,j:|y_i - y_j| < d} \ln d_{ij}. \tag{4}$$

Here, K is a positive constant, $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ is the distance between nodes i and j, and d is the *range* of the repulsive force. This range constant is worth further discussion. The repulsive force is responsible to prevent two nodes from becoming too close in space. The y-coordinates of the nodes have been already fixed, so there is no need to apply a repulsive force between nodes that lie far apart along the y-axis. Correspondingly, we define a non-negative range, d, to the repulsive force, such that two nodes i and j experience mutual repulsion only if $|y_i - y_j| < d$. In practice, the introduction of d accelerates the computation, since the number of mutually repulsive nodes decreases.

Had we been using the gradient to determine the sign of the force, we could have set the extent of the move as a time decreasing temperature function, like in the original model of Fruchterman and Reingold. However, having only two possible directions, we can afford testing more than one possible move for each node. We renounce the computation of the direction, and simply check what would be the Fruchterman-Reingold energy if we move a node a few steps to the right and a few steps to the left. Eventually, we move the node to the location where the energy was the smallest. Formally, if x_i is the location of the i'th node, we calculate the energy in all the $2m + 1$ points $r_i = \{x_i + k \cdot T\}_{k=-m}^{k=m}$, with T the usual time decreasing temperature. The set of points r_i is called the range of x_i. The full algorithm is depicted in Figure 5.

Since the initial vector of coordinates already produces nice drawing, and all that we need is a local smoothing, it suffices to start with small temperature, and to go through a relatively small number of iterations.

4 Further Implications of the y-Axis Arrangement

Here we concentrate on four issues. In the first two we show how our definition of digraph enables to expand standard definitions of familiar terms from the theory of graphs — regular graphs and symmetric nodes. Next we present a definition of an index that measures the amount of hierarchy in a given digraph, and demonstrate how it works. Finally, we discuss the way our algorithm draws cyclic digraphs.

4.1 Regular Digraphs

A *regular graph* is one in which all nodes have the same degree. In analogy, a *regular digraph* (say, for the time being, an unweighted digraph) is one in which all nodes have the

same in-degree and the same out-degree. A regular digraph exhibits high level of symmetry, so that we do not expect to find much hierarchy in it. Indeed, in every digraph the sum of all in-degrees is equal to the sum of all out-degrees. Thus, in a regular digraph each node is balanced, having its in-degree equals to its out-degree. The optimal arrangement, therefore, is the zero vector, telling us to assign the value zero as the y-coordinate of all the nodes. This discussion leads to the following definition of a regular digraph:

Definition 5 (Regular Digraph). *Let $G(V, W, \Delta)$ be a digraph. It will be called a regular digraph if its balance is the zero vector, i.e., if all its nodes are balanced.*

Obviously, this definition generalizes the one we have brought above for unweighted digraphs.

Two examples of regular digraphs are brought in Figure 6. The digraph in 6(a) is an unweighted digraph, and is evidently regular. The digraph in Figure 6(b) is weighted, but still regular.

The optimal arrangement for a regular digraph will obviously be the zero vector $y^\star = (0, \ldots, 0)^T$. In other words, a regular digraph is a hierarchy-free digraph. No directionality is associated with it. Interestingly, any undirected graph is a regular digraph. This is consistent with the above intuition, since undirected edges cannot impose any directionality.

4.2 Symmetric Nodes

A regular digraph is hierarchy-free. Putting differently, when all the nodes are symmetric, they are all assigned with the same y-coordinate. Interestingly, this observation can be further extended. In the following we show that if two nodes are symmetric, then they have the same y-coordinate.

In the framework of undirected graphs, it is customary to denote two nodes i and j as symmetric if there exists a permutation π such that $\pi(i) = j$ and $\pi(j) = i$, and the Laplacian is invariant under π, $L = L^\pi$. Here, L^π is the Laplacian whose rows and columns were permuted according to π.

This definition can be extended to digraphs, by imposing symmetry also on the directionality:

Definition 6 (Symmetric Nodes). *Two nodes i and j are called symmetric if there exists a permutation π such that $\pi(i) = j$ and $\pi(j) = i$, and both the Laplacian and the balance are invariant under π,*

1. $L = L^\pi$.
2. $b = b^\pi$.

b^π is the balance vector whose entries were permuted according to π. This definition reduces to the standard one for undirected graphs, since in this case b is the zero vector.

We expect symmetric nodes to have the same level of hierarchy, i.e., to have identical y-coordinates. This is indeed the case:

Proposition 3. *Let i and j be two symmetric nodes. Then, $y_i^\star = y_j^\star$.*

Proof. Let $y^\star$ be the optimal arrangement obtained from $Ly^\star = b$, and let $y_\pi^\star$ be the optimal arrangement obtained from $L^\pi y_\pi^\star = b^\pi$. Then, since the optimal arrangement is unique, we must have $y_\pi^\star = y^\star$ and in particular

$$(y_\pi^\star)_i = y_i^\star. \tag{5}$$

Function 1D_Force_Model (x, y, G)
% x — the (output) vector of x-coordinates
% y — the pre-computed y-coordinates
% G — the n-node digraph to be plotted

$T \leftarrow$ initial temperature
$L \leftarrow$ list of the nodes sorted by their y-coordinate

while $T >$ final temperature
  % Loop on nodes
  for $i = 1, \ldots, n$
    % E is a vector containing in the l'th place the energy change caused by
    % moving the node to location $x_i + l \cdot T$.
    $E \leftarrow 0$

    % Attractive forces:
    for every j which is adjacent to i
      for every $k = -m, \ldots, m$ % typically, $5 \leqslant m \leqslant 10$
        $d_{ij} \leftarrow \sqrt{(x_i - x_j + k \cdot T)^2 + (y_i - y_j)^2}$
        $E[k] \leftarrow E[k] + d_{ij}^3$
      end for
    end for
    $E \leftarrow E/3K$

    % Repulsive forces (limited by the range d):
    for every j such that $|y_i - y_j| < d$ % Using L
      for every $k = -m, \ldots, m$
        $d_{ij} \leftarrow \sqrt{(x_i - x_j + k \cdot T)^2 + (y_i - y_j)^2}$
        $E[k] \leftarrow E[k] - K^2 \cdot \ln d_{ij}$
      end for
    end for

    $k_{\min} \leftarrow \arg\min_k E[k]$ % Find minimum
    $x[i] \leftarrow x[i] + k_{\min} \cdot T$ % Update coordinates
  end for
  $T \leftarrow T \cdot t$ % Decrease Temperature by a constant factor $t < 1$
end while

Fig. 5. The one-dimensional force directed algorithm. Here, d is the range of the repulsive force, and r_i is the range of the i'th node.

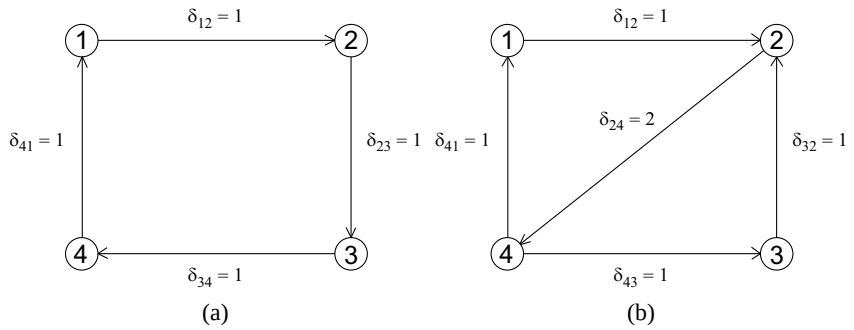


Fig. 6. Two examples of regular digraphs. For both we assume a default weights matrix, W^0.

But a permutation is nothing but a re-naming of the nodes, and therefore

$$(y^\star_\pi)_i = y^\star_j. \tag{6}$$

Combining (5) and (6) we get $y^\star_i = y^\star_j$. □

We would like to emphasize that this proof strongly relies on the uniqueness of the optimal arrangement. This key property of our hierarchy energy enables us to relate symmetry of nodes to actual properties of the drawing.

4.3 Hierarchy Index

The y-axis in our drawings contains the entire available information on the hierarchy. We claim that the spread of the projection of the drawing on this axis is closely related to its inherent hierarchy. Two extreme examples are shown in Figure 7. In Figure 7(a) a circle is shown. Clearly, no node is different from the other, and we do not expect to see any amount of hierarchy. Indeed, it is a regular digraph, thus $y^\star = (0, \ldots, 0)^T$. In Figure 7(b), a path is shown. There, the amount of hierarchy is maximal, and indeed each node has a different y-coordinate in constant increments, $y^\star = (2, 1, 0, -1, -2)^T$.

It would be natural, therefore, to associate the hierarchy of a digraph with the magnitude $\Delta y^\star = y^\star_{\max} - y^\star_{\min}$. The larger is $\Delta y^\star$, the more hierarchical is the digraph.

One can use this magnitude to measure how worthwhile it is to allot the y-axis to exhibit the directional information. In order to do so, $\Delta y^\star$ should be compared with a measure of the dimension of the graph, had it was drawn using undirected graph drawing algorithms. A plausible candidate for measuring this is the diameter of the digraphs, D, which is the graph theoretic distance between the farthest nodes[1]. Therefore:

Definition 7 (Hierarchy Index). *Let $G(V, W^0, \Delta^0)$ be an unweighted digraph. Its hierarchy index is*

$$H = \frac{\Delta y^\star}{D} = \frac{y^\star_{\max} - y^\star_{\min}}{D},$$

where $y^\star$ is its optimal arrangement and D its diameter.

[1] As a matter of fact, $\Delta y^\star$ and D can be compared only for unweighted digraphs, and for this reason we assume in this subsection only such digraphs. It is not difficult to generalize the definition of the diameter, such that $\Delta y^\star$ and D could be compared also for weighted digraphs, but we will not do it here.

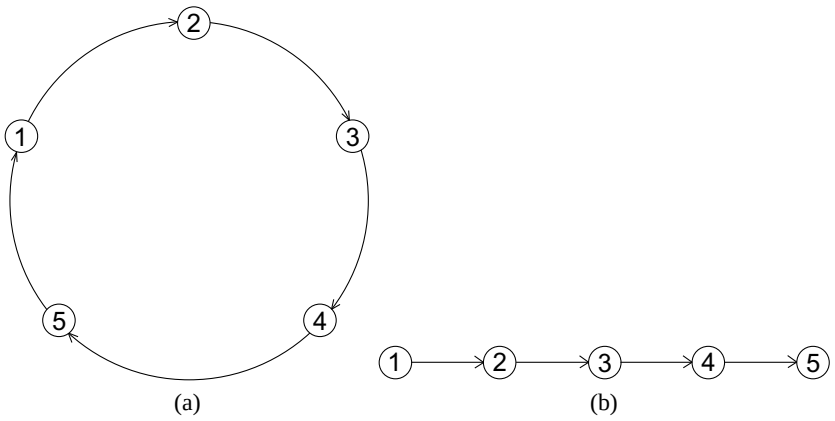


Fig. 7. The digraph in (a) is with minimal amount of hierarchy, while that in (b) is of maximal amount of hierarchy.

If $\Delta y^\star$ is comparable with D, then the directional information is significant and one should use digraph drawing algorithms. If, on the other hand, $\Delta y^\star$ is small with respect to D, then it is not "profitable" any longer to dedicate an entire axis for the directional information, and undirected graph drawing algorithms should be preferred.

Regular digraphs are an extreme case of the later scenario. Here $\Delta y^\star = 0$, and their directed drawing is essentially one dimensional, making no use of the y-axis. It is recommended, in this case, to use undirected graph drawing algorithms. For example, one can adopt Hall's idea and to use for the y-axis the Fiedler vector instead of the optimal arrangement. In Figure 8 we see how the y-axis would look like in this case, for the two regular digraphs of Figure 6. In Figure 8(b) we see the arrangement imposed by the Fiedler vector of the digraph from Figure 6(b), $(1, 0, -1, 0)^T$. The Fiedler vector of the digraph from Figure 6(a) is not unique, since the smallest positive eigenvalue of the Laplacian is two-fold degenerate. Figure 8(a) was obtained by choosing the Fiedler vector $(1, 0, -1, 0)^T$ to represent the layering.

We shall next see some examples for the hierarchy index:

1. **Regular digraph**: For any regular digraph, for example undirected graphs and circles, $H = 0$.
2. **Binary tree**: The diameter of a complete binary tree with n nodes is $2 \log n$. The magnitude $\Delta y^\star$ is, $\log n$, see also section 5. The hierarchy index of a complete binary tree is therefore $H = \frac{1}{2}$, independent of n. This 1:2 ratio is well visualized when comparing the height of the hierarchical drawing, as in Figure 14(b), with that of the radial drawing generated by undirected force models, as in Figure 9.
3. **Path**: The diameter of an n-node path is $n - 1$ (see for example the 5-node path of Figure 7(b)). $\Delta y^\star$ is also $n - 1$. Consequently, the hierarchy index of a path is $H = 1$, independent of n.
4. **Circle with extension**: Let one of the nodes of an n-node circle be pushed by one node outside the circle, see example in Figure 10. The diameter of such a digraph is roughly $n/2$. The magnitude $\Delta y^\star$ is 1. Therefore, $H \approx \frac{2}{n}$. As expected, $\lim_{n\to\infty} H = 0$.

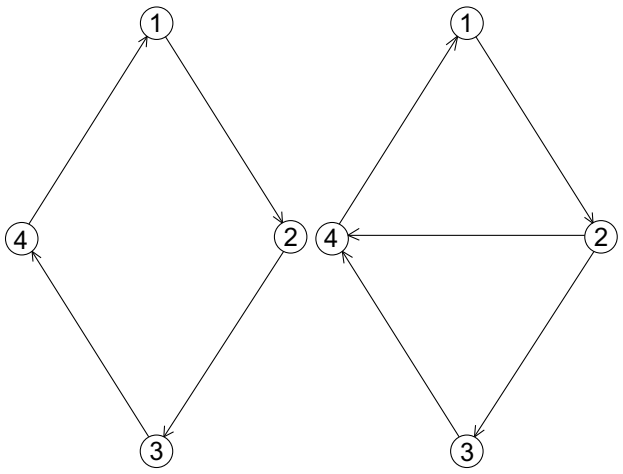


Fig. 8. The layering of regular graphs, as resulting from using the Fiedler vector.

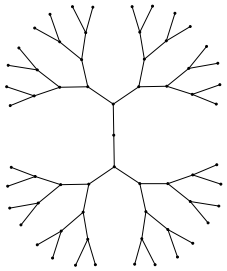


Fig. 9. An undirected force model drawing of a binary tree.

4.4 Cyclic Digraphs

Standard layering algorithms can be applied only to acyclic digraphs. When having to deal with cyclic digraphs, they are first transformed into acyclic ones by inverting the direction of a minimal number of edges. Our algorithm allows to directly draw cyclic digraphs, without having to invert edge directions. We believe it to be one of its most prominent advantages.

To make this claim stronger we here show why it seems that there is no simple connection between the number of edges whose direction should be reversed, and the inherent hierarchy of the digraph. For example, in a circle like the one plotted in Figure 7(a), it suffices to invert the direction of a single edge in order to make it acyclic. Thus, it will be drawn by standard layering algorithms as a full-hierarchy path, having the lowest and the highest nodes connected by a reversed edge, see Figure 11. Obviously, this mis-represents the structure of the hierarchy-free circle. Applying our algorithm to a directed circle shows that it contains no directionality, being a regular digraph. In the absence of hierarchy, there is no sense in forcing the edges to be all in the same direction.

Another example is shown in Figure 12. Here, the digraph does contain hierarchy, and the figure shows its optimal arrangement as dictated by our algorithm. We believe that we

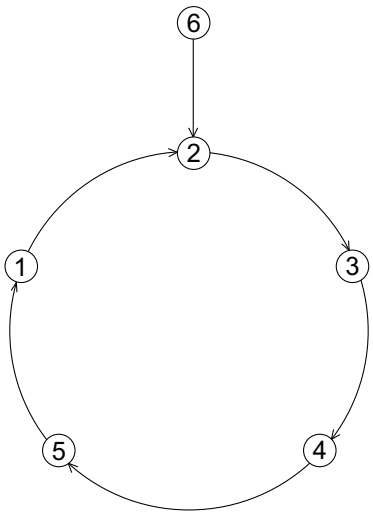


Fig. 10. A 6-node circle with extension.

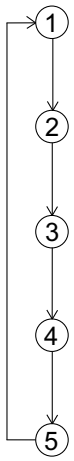


Fig. 11. Schematic layering of 5-points circle.

can quite objectively claim that this drawing best represents the structure of this digraph, despite of the fact that about half of the edges point downwards, and about half point upwards. This is because the only explicit hierarchy in this digraph, which is well captured in the figure, is between the highest node and the lowest one. All the other nodes do not possess evident hierarchical relations, thus some of the edges connecting them are "allowed" to go upwards.

5 Examples

We have been successfully testing our algorithm against several digraphs with diverse structures. Here, we provide layouts of several unweighted digraphs. All the drawings were ob-

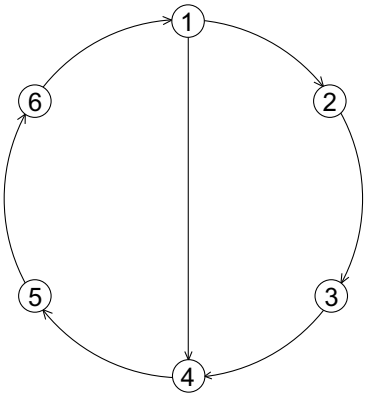


Fig. 12. Schematic optimal arrangement of a variation of a circle. This variation, unlike a circle, does contain hierarchy.

tained using the minimum linear arrangement solution as the first stage of computing the x-coordinates. For all the examples, the computation time was instantaneous.

Figure 13 shows three instructive examples. Figure 13(a) shows the actual drawing of the digraph we have presented in Section 4.4. This drawing is identical in structure with the schematic one from Figure 12. In contrast to a circle, which is regular and thus hierarchy-free, this digraph, thanks to the extra edge, does contain hierarchical information. Figure 13(b) shows an acyclic digraph comprised of a few parallel paths of different lengths. In spite of the diversity of path lengths, all edges are drawn in the same direction. Figure 13(c) is a cyclic version of the former digraph, with the direction of the edges along one of the paths being inverted. Interestingly, the drawing is almost identical to that of the acyclic version, with the principal difference being the direction of the "reversed" edges.

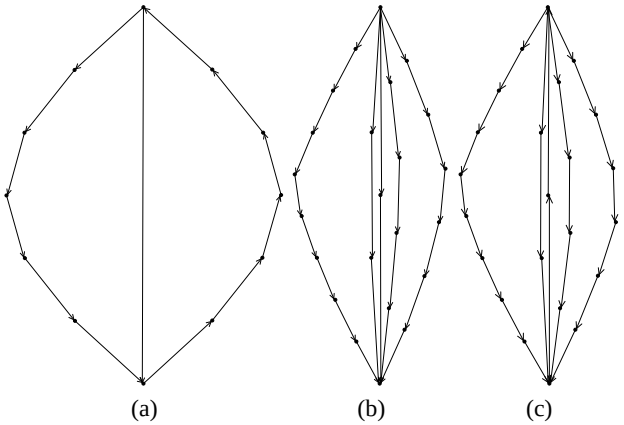


Fig. 13. Three examples of digraphs. **(a)** A distorted circle. The extra edge is responsible for the observed hierarchy. **(b)** An acyclic digraph comprised of a few parallel paths with varying lengths. **(c)** A cyclic version of the former digraph.

Figure 14 shows a complete 5-level binary tree. The y-coordinates were naturally quantized into 6 layers, as dictated by the tree structure. Recall, that assigning the x-coordinates is a two-stage algorithm. Figure 14(a) shows the drawing as obtained after the first stage, and Figure 14(b) shows the drawing as obtained after both stages. In this example the global structure of the digraph was already captured by the first stage, and the second stage does only local beautification. This behavior is not specific for this example, but is the general rule. To obtain 14(c), we have inserted an edge from a central leaf to the root, thus forming a circular path in the binary tree. As there is nothing in the digraph structure that gives a precedence to one of the nodes in this path, we expect them all to have the same level of hierarchy. The drawing reflects this observation, putting these nodes at the highest level.

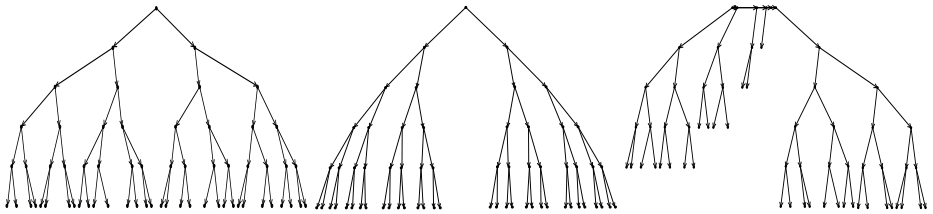


Fig. 14. (a) A 5-level complete binary tree, the x-coordinates obtained using only the first stage of the algorithm. **(b)** The same digraph, using both stages for producing the x-coordinates. **(c)** Closing a directed circle in (a), by inserting an edge from a central leaf to the root.

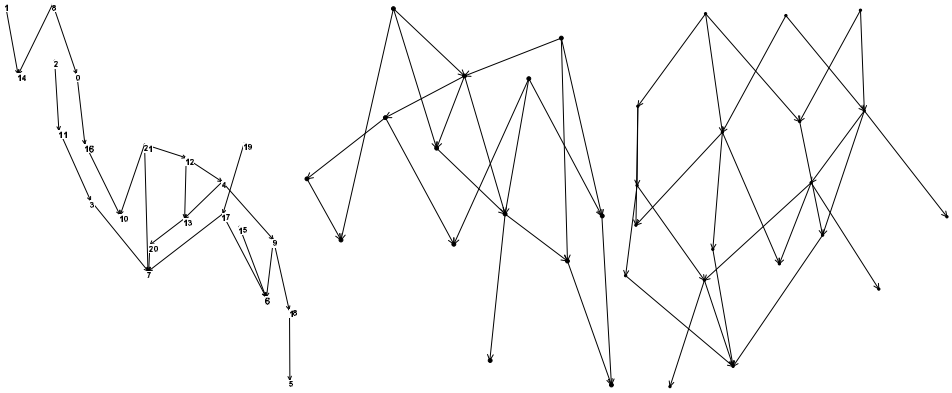
Figure 15 shows three digraphs that have appeared in literature, and the reader is invited to compare our results with those shown in the original works. Figure 15(a) shows an example that was analyzed in details in Figure 1 of [10]. There, the authors obtained the drawing of this digraph using four different variations of standard layering algorithms. Figure 15(b) shows an example that was manually illustrated in [1], page 487. In Figure 15(c) we bring our drawing of the digraph shown in Figure 9.1 of [3]. All these examples demonstrate the way our algorithm captures the structure of the digraphs. A salient difference between our drawings and their standard layering counterparts, is that the nodes are not arranged in strict layers, but rather placed in a continuous fashion, that reflects their unique hierarchical status.

Finally, we bring in Figure 16 two digraphs that describe partial order of system execution, as generated by the play engine tool for requirement specification and execution [9]. These examples are interesting, because they describe common execution constructs, such as loops and conditions.

6 Discussion

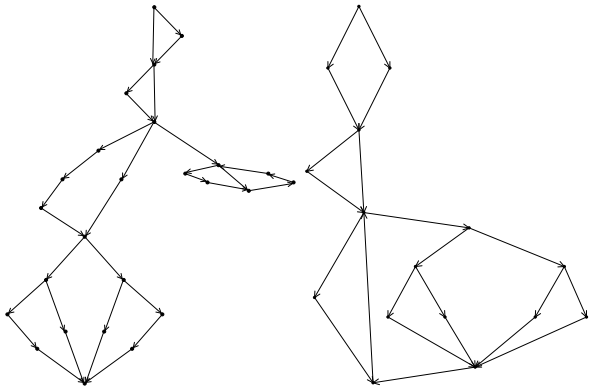
We have presented a digraph drawing algorithm that uses a sequence of three energy minimization problems (or, in other words, three force directed models) to find an optimal drawing in two-dimensions.

The vector of y-coordinates, namely the optimal arrangement, is found using an elegant energy minimization algorithm, that yields a unique global minimizer. Moreover, as



(a) (b) (c)

Fig. 15. Three examples of digraphs that have appeared in literature. See citation details in text.



(a) (b)

Fig. 16. Two examples that show partial order of system execution.

we shall shortly discuss in more details, it captures in a profound way the directional information of the digraph.

For the vector of x-coordinates, which contains no directional information, we have shown that traditional gradient descent optimization algorithms fail when restricted to our one dimensional case. Instead, we have developed an optimization algorithm tailored especially for the one dimensional problem. This algorithm, which solves two energy minimization problems in a row, converges much faster than two-dimensional force directed algorithms.

In our eyes, two of the most important achievements of our algorithm is its ability to draw cyclic digraphs without having to invert edge directions, and its ability to measure the amount of hierarchy in the digraph via the hierarchy index. This last property can be used to decide whether to use hierarchical drawing tools to represent a given digraph, or should we prefer undirected graph drawing algorithms.

This last issue calls for an interesting lead that we believe worth further research. We suggest the possibility to combine digraph drawing algorithms, and undirected graph draw-

ing algorithms into a unified tool — given a general digraph, we might use the hierarchy index on portions of it, and draw the different portions either with this algorithm or with the other, depending of their level of hierarchy. More specifically, one can scan the optimal arrangement vector to find connected subgraphs, such that the nodes in each subgraph have similar y-coordinates. Such subgraphs are suspected to form undirected components, and should be processed separately to decide what are the best tools to draw them.

Our algorithm can be used in two different ways for the benefit of the standard approach for digraph drawing:

- **It can induce layering**: We can think of the optimal arrangement as a kind of a "continuous layering". The usual discrete layering can be easily induced from it, if we divide the nodes into maximal subsets, such that within each subset the nodes will have successive y-coordinates, and such that no edge resides within a single subset.
- **It can induce ordering**: Standard ordering algorithms are normally very local in nature. In a single iteration only one layer is free to change the order of its nodes. We can replace it with a more global approach, using the vector of x-coordinates obtained by our "first stage" to impose a "global ordering".

Finally, we would like to mention some properties of graphs that we have defined, or actually re-defined. For one, there is the target height differences matrix, which had proved itself to be of great value. Second, the notions of regularity and symmetry were generalized, and related to properties of the drawing itself.

References

1. T. H. Cormen, C. E. Leiserson and R. L. Rivest, *Introduction to Algorithms*, MIT Press, 1990
2. R. Davidson and D. Harel, "Drawing Graphs Nicely Using Simulated Annealing", *ACM Trans. on Graphics* **15** (1996), 301-331.
3. G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis, *Algorithms for the Visualization of Graphs*, Prentice-Hall, 1999.
4. J. Diaz, J. Petit and M. Serna, "A Survey on Graph Layout Problems", Technical report LSI-00-61-R, Universitat Politècnica de Catalunya, Departament de Llenguatges i Sistemes Informàtics, 2000. To appear in *ACM Computing Surveys*.
5. P. Eades, "A Heuristic for Graph Drawing", *Congressus Numerantium* **42** (1984), 149-160.
6. T. M. G. Fruchterman and E. Reingold, "Graph Drawing by Force-Directed Placement", *Software-Practice and Experience* **21** (1991), 1129-1164.
7. G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, USA, 1996.
8. K. M. Hall, "An r-dimensional Quadratic Placement Algorithm", *Management Science* **17** (1970), 219-229.
9. D. Harel and R. Marelly, "Specifying and Executing Behavioral Requirements: The Play-In/Play-Out Approach", Technical Report MCS01-15, The Weizmann Institute of Science, Faculty of Math. and Computer Science, 2001.
10. P. Healy and N.K. Nikolov, "How to layer a directed acyclic graph", *Proceedings of Graph Drawing 2001*, Lecture Notes in Computer Science, Vol. 2265, pp. 16–30, Springer Verlag, 2001.
11. T. Kamada and S. Kawai, "An Algorithm for Drawing General Undirected Graphs", *Information Processing Letters* **31** (1989), 7-15.
12. M. Kaufmann and D. Wagner (Eds.), *Drawing Graphs Methods and Models*, Lecture Notes in Computer Science, Vol. 2025, Springer Verlag, 2001.

13. Y. Koren, L. Carmel and D. Harel "ACE: A Fast Multiscale Eigenvectors Computation for Drawing Huge Graphs", Technical Report MCS01-17, The Weizmann Institute of Science, Faculty of Math. and Computer Science, 2001. Available at: `www.wisdom.weizmann.ac.il/reports.html`.
14. Y. Koren and D. Harel "A Multi-Scale Algorithm for the Linear Arrangement Problem", Technical Report MCS02-04, The Weizmann Institute of Science, Faculty of Math. and Computer Science, 2002. Available at: `www.wisdom.weizmann.ac.il/reports.html`, to appear in *Proceeding of Graph Theoretical Concepts in Computer Science 2002*, Springer-Verlag.
15. K. Sugiyama and K. Misue, "A Simple and Unified Method fro Drawing Graphs: Magnetic-Spring Algorithm", *Proceedings of Graph Drawing 1994*, Lecture Notes in Computer Science, Vol. 894, pp. 364–375, Springer Verlag, 1995.
16. K. Sugiyama, S. Tagawa and M. Toda, "Methods for Visual Understanding of Hierarchical Systems", *IEEE Transactions on Systems, Man, and Cybernetics* **SMC-11(2)** (1981), 109–125.
17. W. T. Tutte, "How to draw a graph", *Proceedings London Mathematical Society* **13** (1963), 743-768.