Combining Hierarchy and Energy for Drawing Directed Graphs

Liran Carmel, David Harel, *Fellow, IEEE,* and Yehuda Koren

***Abstract*— We present an algorithm for drawing directed graphs, which is based on rapidly solving a unique one-dimensional optimization problem for each of the axes. The algorithm results in a clear description of the hierarchy structure of the graph. Nodes are not restricted to lie on fixed horizontal layers, resulting in layouts that convey the symmetries of the graph very naturally. The algorithm can be applied without change to cyclic or acyclic digraphs, and even to graphs containing both directed and undirected edges. We also derive a hierarchy index from the input digraph, which quantitatively measures its amount of hierarchy.**

***Index Terms*— Directed graph drawing, force directed layout, hierarchy energy, Fiedler vector, minimum linear arrangement.**

I. Introduction

VISUALIZING directed graphs (digraphs) is a challenging task, requiring algorithms that faithfully represent the relative similarities of the nodes, as well as to give some sense of the overall directionality. The latter requirement renders algorithms designed for undirected graphs inappropriate for 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.* [21], is based on separating the axes, where the y-axis represents 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.

In these Sugiyama-style algorithms, the y-coordinates are computed by dividing the y-axis into a finite number of layers and associating each node with exactly one layer — a process called *layering*. Edges are allowed only between consecutive layers, and they all point in the same direction. Whenever an edge is about to cross a layer, a *dummy node* is inserted to prevent this. When dealing with cyclic digraphs, no layout can place all edges in the same direction, and what is traditionally done in this case is to apply a preliminary stage, in which a minimal number of edges is sought, whose reversal will make the digraph acyclic. This is actually an NP-hard problem, but several sub-optimal algorithms have been proposed. 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 two adjacent layers are considered: the order of the nodes is kept fixed in one, and is determined in the other layer in order to reduce the number of edge crossings. This too is an NP-hard problem. 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. For more details see [3], [10], [13].

Such digraph drawing algorithms have evolved to produce nice and useful layouts for many different types of digraphs. Nevertheless, we would like to point out two inherent properties of the standard strategy, which, despite being treated in various ways by the many algorithms, are in many cases still undesirable:

- Finding a layering for cyclic digraphs requires transforming them into acyclic ones, thus introducing a certain distortion of the original problem.
- 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[1]. It embraces the idea of axis separation, but uses novel approaches to the drawing of both axes. These approaches, apart from the fact that they produce nice drawings and have fast implementations, also successfully deal with the two aforementioned points — the distortion and the discrete layering.

We associate with the nodes continuous y-coordinates, in a way that suggests a natural unified framework that can be applied to any kind of digraph, whether cyclic or acyclic, and which requires no graph modification or preprocessing. 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, in order to better represent the hierarchy. If strict layering is nevertheless important, it can be easily induced from the continuous y-coordinates by a simple quantization process.

We define the vector of y-coordinates as the unique minimizer of a simple energy (cost function), and show that the minimization problem is equivalent to a system of linear equations, whose solution can be found with high speed. This energy function, to which we call the *hierarchy energy*, strongly reflects the directional information of the digraph. Its simple form enables rigorous analysis, giving rise to many interesting results, the most important of which appears to be the definition of an index for measuring 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. Thus, they are

[1] An early and short version of this work appeared in: L. Carmel, D. Harel and Y. Koren, "Drawing Directed Graphs Using One-Dimensional Optimization", *Proc. Graph Drawing (GD'02)*, LNCS 2528, pp. 193–206, Springer-Verlag, 2002.

simultaneously assigned (without having to go through an iterative process) using the minimizer of another energy function that is suitable for the one-dimensional case. We suggest three alternatives for such an energy function. Surprisingly, despite of the fact that no directional information is expressed by the x-axis, the most successful of these energy functions strongly relates to the same hierarchy energy used for calculating the y-coordinates.

Had we been asked to categorize our algorithm, we would have said that it is purely energy minimization oriented, as all of its parts use energy minimization procedures, each part with its own specially tailored energy function. By definition, a force is the inverse gradient of the energy. Thus, the strategy of energy minimization is equivalent to a force directed model. Force directed models are much more popular in undirected graph drawing than 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 [2], [11]. We are aware of only two other occasions where a force directed model was suggested for digraph drawing, [12], [20], but in both cases we are under the impression that the inferred energy function is complicated, rich in local minima, and rather difficult to minimize.

II. Basic Notions

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. Henceforth, 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.

Each edge $\langle i, j \rangle$ is associated with a *weight* w_{ij}. We assume that the weights are always non-negative ($w_{ij} \geqslant 0$ for all $\langle i, j \rangle$), and that there are no self-edges ($w_{ii} = 0$ for all i). For simplicity of notation we shall use the convention that $w_{ij} = 0$ for any non-adjacent pair. We symbolize by W the corresponding $n \times n$ matrix of weights. Obviously, this is a symmetric matrix, $w_{ij} = w_{ji}$.

We further associate with each edge (i, j) a number δ_{ij} that measures the desired height difference between nodes i and j along the y-axis, thus expressing the relative hierarchy of the nodes. Accordingly, in the drawing we would like to place nodes i and j such that $y_i - y_j = \delta_{ij}$. We shall call these numbers as *target height differences*, and use the convention $\delta_{ij} = 0$ for any non-directed edge, or when there is no edge between nodes i and j. We symbolize by Δ the corresponding $n \times n$ 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 $G(V, E; W, \Delta)$. 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 $w_{ij} = \delta_{ij} = 1$. Hereinafter, we shall term a digraph with these default values an *unweighted digraph*, and denote its weights matrix and target height differences matrix by 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 entities associated with a digraph $G(V, E; W, \Delta)$ — the *Laplacian* and the *balance*:

Definition 1 (Laplacian): Let $G(V, E; W, \Delta)$ 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 between digraphs and undirected graphs. The Laplacian has a key role in some undirected graph drawing algorithms; see, e.g., [9], [14], [16], [17], and will be shown to play a fundamental role here, too. One of its most important properties is the following:

Lemma 1: Let $G(V, E; W, \Delta)$ be a digraph. Its Laplacian is a positive semi-definite matrix, and thus has 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 $1_n = (1, \ldots, 1)^T \in \mathbb{R}^n$ and c any constant.

Proof: See Hall [9]. ∎

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

Definition 2 (Balance): Let $G(V, E; W, \Delta)$ 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*. A digraph all of whose nodes are balanced will be called a *balanced digraph*.

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

Lemma 2: Any balance vector b is orthogonal to the vector 1_n, i.e.,

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

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. ∎

III. The Algorithm

In this section we define the hierarchy energy and develop the theory that underlies the different parts of the drawing algorithm. Some details of implementation are postponed to Section IV.

A. Assigning the y-Coordinates

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

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

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

Clearly, $E_H(y) \geq 0$ 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, $y^\star = \arg\min_y E_H(y)$.[2] 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 $(y_i - y_j - \delta_{ij})^2$ be small. The larger this quantity, the smaller $(y_i - y_j - \delta_{ij})^2$ should be, in order to keep the contribution to the energy small. A similar energy function was used by Brandes *et al.* [1] for a completely different drawing application (edge routing of timetable graphs).

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

Lemma 3: Let $G(V, E; W, \Delta)$ 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

$$E_H(y) = E_0 + y^T L y - 2y^T b, \quad (2)$$

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

Proof: Expanding the hierarchy energy (1), we get

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

The first term was shown in [9] 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.

[2] $x_0 = \arg\min_x f(x)$ is a notation commonly used to describe the minimizer x_0 of $f(x)$. This is to be distinguished from $\min_x f(x)$ which is just $f(x_0)$.

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

$$Ly = b.$$

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

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

Thus, the solution of $Ly = b$ corresponds to an extremum of $E_H(y)$. It is a global minimum, since $E_H(y)$ is 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 $G(V, E; W, \Delta)$ 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: The existence of at least one solution follows from the fact that the energy $E_H(y)$ 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 follows 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 V, this exact property is the one that makes feasible the definition of a 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 at the origin of the coordinates, i.e., $\sum_i y_i^\star = 0$. This choice of $y^\star$ enables its fast computation using the *conjugate gradient method*; see Section IV. Therefore, we redefine the optimal arrangement as:

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

Let us see how this algorithm works by applying it to some very small-scale examples. More examples appear in later sections. Figure 1(a) shows an unweighted acyclic digraph. Its optimal arrangement is the solution of the system

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

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

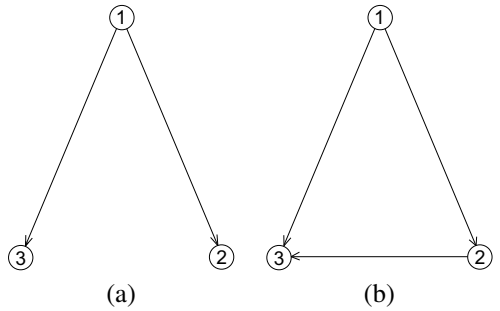


Fig. 1. Two very small examples of unweighted acyclic digraphs.

Figure 1(b) shows another example of an unweighted acyclic digraph. In this case,

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

which is the constrained solution of the system

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

Aesthetically, this vector of coordinates nicely captures the structure of the digraph, where, in contrast to the first 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 compromise of sorts, pushing node 2 upwards and node 3 downwards, thus decreasing the height difference $y_1 - y_2$ to $\frac{2}{3}$ and increasing $y_1 - y_3$ to $\frac{4}{3}$. The height differences cannot achieve their targets, resulting in a strictly positive hierarchy energy $E_H(y^\star) = (\frac{2}{3} - 1)^2 + (\frac{2}{3} - 1)^2 + (\frac{4}{3} - 1)^2 = \frac{1}{3}$.

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

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

giving

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

which is schematically plotted in Fig. 2(b). Here we see the naturalness of the way 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, resulting 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 on one axis of the drawing, was already exploited in the field of undirected graph drawing by Tutte [22] and Hall [9], 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.

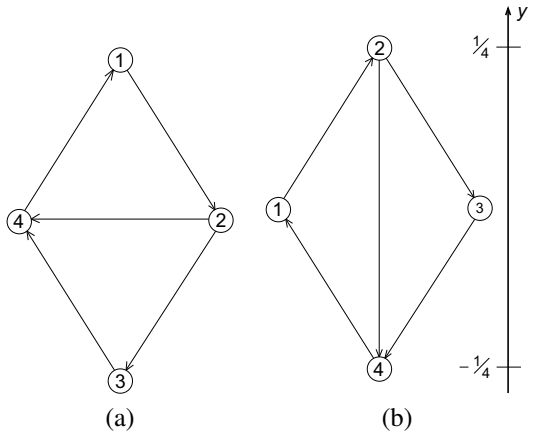


Fig. 2. **(a)** A small example of an unweighted cyclic digraph; **(b)** its optimal arrangement.

Tutte and Hall used the following quadratic energy function:

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

Comparing this energy with the hierarchy energy (1), it is clear that they become identical for a digraph with $\Delta = 0$, which is really an undirected graph. Furthermore, undirected graphs are members of the larger family of balanced digraphs, 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 V, and here we just briefly explain how Tutte and Hall dealt with it in the framework of undirected graph drawing:

- **Tutte's solution**: Tutte [22] arbitrarily chose a certain number of nodes to be anchors, i.e., he fixed their coordinates in advance. This, of course, prevents the collapse of all nodes to the same location, but instead it raises new problems, such as which nodes should be the anchors, and how to determine their coordinates.
- **Hall's solution**: Hall [9] 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 the examples in [9] and [16].

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 $\partial E_{TH}(y)/\partial y_i = 0$ 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 the equation $\partial E_H(y)/\partial y_i = 0$ which gives

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

This yields 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 the barycenter. The introduction of nonzero δ_{ij}'s prevents the collapse of all the nodes to the same location, yielding a meaningful solution.

B. 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'. Recall that the y-coordinates of the nodes are already fixed, and now the nodes are allowed to move only along the x-axis. However, if two nodes have close y-coordinates, swapping places along the x-axis is almost always impossible, even if it is energetically favorable, due to the repulsive forces that form an "energy barrier". 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 ones, thus preventing further progress of the nodes.

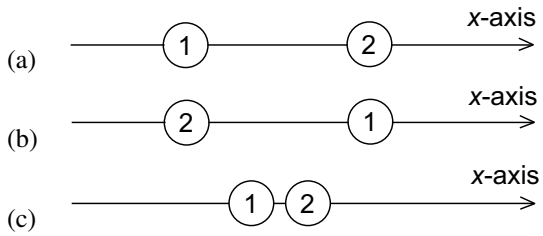


Fig. 3. Visualization of the swapping problem.

It would be the best, then, to employ an alternative optimization technique for our one-dimensional case, which skirts the swapping problem by avoiding the node-by-node optimization mechanism. This calls for associating the vector $x = (x_1, \ldots, x_n)^T$ representing the x-coordinates of the nodes with an energy function that can be minimized using global (as opposed to node-by-node) techniques. To this end, we would like to suggest three such energy functions, each is driven by different aesthetic reasoning:

Minimizing edge-squared lengths: This means minimizing the already familiar Tutte-Hall energy function, $E_{TH}(x) = \frac{1}{2}\sum_{i,j=1}^n w_{ij}(x_i - x_j)^2 = x^T L x$. As discussed in Subsection III-A; see also [14], 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 [16] — an extremely fast multiscale algorithm for undirected graph drawing.

Minimizing edge lengths: This is the well known problem of *minimum linear arrangement* [4]. The solution is obtained by minimizing the energy function $E_{LA}(x) = \frac{1}{2}\sum_{i,j=1}^n w_{ij}|x_i - x_j|$, where $(x_1, \ldots, x_n)$ is a permutation of $(1, \ldots, n)$. This is an NP-hard problem, and hence we should work with heuristics. In practice, we find a local minimizer of E_{LA} using another fast multiscale algorithm [18], designed especially for this problem.

Minimizing the stress: This means minimizing the stress energy, $E_S(x) = \frac{1}{2}\sum_{i,j=1}^n k_{ij}\left(|x_i - x_j| - d_{ij}\right)^2$, where d_{ij} is the graph-theoretical distance between nodes i and j, and $k_{ij} = 1/d_{ij}^2$ is a normalization constant. This energy is a traditional measure of drawing quality, based on the heuristic that a nice drawing relates to good isometry [11]. Its minimization calls for placing the nodes so that the resulting pairwise Euclidean distances will approach the corresponding graph-theoretical distances. Interestingly, the stress energy shares much resemblance to the hierarchy energy (1), with the only difference being the absolute value appearing in the former. In another work [15] it is shown that a local minimizer of the stress energy can be found by sequentially optimizing a series of hierarchy energies. Impressively enough, the hierarchy energy is found to fulfill a valuable role also in determining the x-coordinates, despite of the fact that they does not carry directional information.

Specific examples of the layouts obtained by the three energy functions are shown in Section VI; see Figure 10. Here we would like to keep the discussion more theoretical, and dwell upon some of the pros and cons of the three alternatives. The minimization process of the Tutte-Hall energy function is guaranteed to converge to the global minimum. This is reflected in very smooth layouts that very impressively capture symmetries in the digraph; see Figure 15. Consequently, it seems highly adequate for drawing large and well-connected digraphs with some clear structure. The Tutte-Hall energy minimization is further favorable when very large digraph are concerned, since the solution can be found extremely fast using multiscale techniques; see [16], [17] and Table I. In other cases (most notably, trees), 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 two nodes from having the same x-coordinate. Therefore, locally dense regions could very well appear. Using the minimum linear arrangement for the x-coordinates solves this last drawback of the Fiedler vector. It allocates a different integral x-coordinate per node, thus preventing overcrowding of the nodes. However, the final result might in some cases be somewhat "unnatural" due to this quantization of the x-axis. Using the stress energy

minimization yields, in many cases, the most "natural" results, with the overall Euclidean distances between the nodes highly correlated with their graph-theoretical distances.

IV. Optimization of the Hierarchy Energy

Finding the optimal layering means solving the system $Ly = b$; see Definition 4. Classical solvers, however, might fail, since, by Lemma 1, L is singular. However, here we note the fundamental importance of the fact that we have been defining the optimal arrangement to be orthogonal to 1_n. As will be explained shortly, this fact enables us to solve the system $Ly = b$ by direct application of a conjugate gradient algorithm [8]; see Figure 4. All that we have to do is to make sure that the initial vector is orthogonal to 1_n. Convergence is then guaranteed in n iterations, but in practice it 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.

Function Conjugate_Gradient (L, b)
% L — the Laplacian
% b — the balance
const $\epsilon \sim 10^{-3}$ % tolerance

% Initialization:
$y \leftarrow$ choose at random
$y \leftarrow y - \frac{1}{n}\sum_{i=1}^{n} y_i$ % Orthogonalization against 1_n
$d \leftarrow r \leftarrow b - Ly$

% Loop until convergence:
while $\|r\| > \epsilon$ **do**
  $\gamma \leftarrow r^T r$
  $\alpha \leftarrow \frac{\gamma}{d^T L d}$
  $y \leftarrow y + \alpha d$
  $r \leftarrow r - \alpha L d$
  $\beta \leftarrow \frac{r^T r}{\gamma}$
  $d \leftarrow r + \beta d$
end while
return y

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

For the interested reader, we now show why we can safely apply a conjugate gradient algorithm to the system $Ly = b$. 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., $P \cdot 1_n = (1, 0, \ldots, 0)^T$. 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, allowing to ignore that axis and to restrict the problem to the $(n-1)$-dimensional subspace orthogonal to 1_n. 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

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

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 can be quickly solved and with excellent numerical stability. If L'' is dense, one can solve the system using Cholesky factorization [8], which is faster and more stable than Gauss elimination. If L'' is sparse, iterative techniques should be preferred. Gauss-Seidel relaxation [8] is guaranteed to converge, but the conjugate-gradient (CG) technique [8] is considered stronger, with a guaranteed 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 in 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 1_n direction. Thus, as long as we start with a vector orthogonal to 1_n, we are guaranteed that the final solution $y^\star$ will be orthogonal to 1_n, too. This justifies our use of the conjugate gradient algorithm directly on $Ly = b$.

V. Further Implications of the y-Axis Arrangement

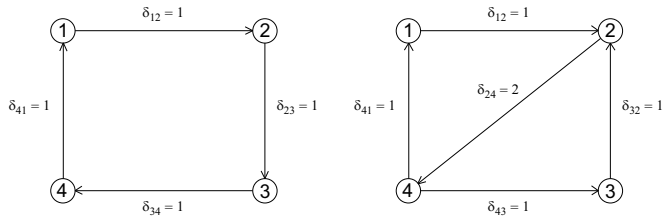
Here we concentrate on four issues. In the first two subsections we characterize the way our algorithm operates on regular graphs and on symmetric nodes. Next we present a definition of an index that measures the amount of hierarchy in a given digraph, and demonstrate how it works. Finally, we discuss the way our algorithm draws cyclic digraphs.

A. Regular Digraphs

A *regular graph* is one in which all nodes have the same degree. In analogy, a *regular digraph* is one in which all nodes have the same in-degree and also all nodes have the same out-degree. A regular digraph exhibits a high level of symmetry, so that we do not expect to find much hierarchy in it. Indeed, our algorithm reflects this absence of hierarchy by placing all nodes at the same y-coordinate. This can be proved from the observation that in a regular digraph *each* node is balanced, having equal in-degree and out-degree (as can be deduced from the fact that in every digraph the sum of all in-degrees is equal to the sum of all out-degrees). Hence, the optimal arrangement for a regular digraph is the solution of $Ly = 0$, which is $y^\star = (0, \ldots, 0)^T$.

In general, the optimal arrangement of any balanced digraph (Definition 2) would be the zero vector $y^\star = (0, \ldots, 0)^T$. Two examples of such digraphs are shown in Figure 5. Interestingly, any undirected graph is a balanced digraph. This is consistent with the above intuition, since undirected edges cannot impose any directionality.

Clearly, using the optimal arrangement for the y-coordinates of regular digraphs is useless, and we shall prefer undirected drawing algorithms.



(a) (b)

Fig. 5. Two examples of balanced digraphs. For both we assume a default weights matrix, W^0.

B. Symmetric Nodes

When the nodes are all symmetric the digraph is regular, hence the nodes are all assigned the same y-coordinate. Interestingly, this observation can be extended: if two nodes are symmetric, 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 are permuted according to π. For digraphs, we impose symmetry also on the directionality:

Definition 5 (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) $b = b^\pi$.

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

$$(y_\pi^\star)_i = y_i^\star. \tag{4}$$

But a permutation is nothing but a re-naming of the nodes, so that

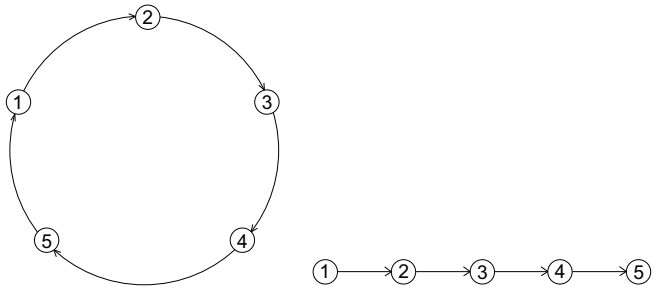
$$(y_\pi^\star)_i = y_j^\star. \tag{5}$$

Combining (4) and (5), we get $y_i^\star = y_j^\star$. ∎

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

C. Hierarchy Index

The y-axis in our drawings contains the entire available information about 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 6. In Figure 6(a) a circle is shown. Clearly, no node is different from the other, and we do not expect to see any hierarchy at all. Indeed, it is a regular digraph, thus $y^\star = (0, \ldots, 0)^T$. In Figure 6(b), a path is shown. There, the amount of hierarchy is maximal, each node has a different y-coordinate in unit increments, $y^\star = (2, 1, 0, -1, -2)^T$.



(a) (b)

Fig. 6. The digraph in (a) is hierarchy-free, while that in (b) has a maximal amount of hierarchy.

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 the $\Delta y^\star$, the more hierarchical 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 digraph, had it been drawn using undirected graph drawing algorithms. A plausible candidate for measuring this is the diameter D of the digraph, which is the graph-theoretical distance between the two most distant nodes. Therefore:

Definition 6: The **hierarchy index** of an unweighted digraph[3] is

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

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

If $\Delta y^\star$ is comparable to D, 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 no longer 'profitable' to dedicate an entire axis for the directional information, and undirected graph drawing algorithms should be preferred.

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 VI. 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 10, with that of the radial drawing generated by undirected force models, as in Figure 7.

[3]It is possible to generalize the definition of the hierarchy index for weighted digraphs, but we will not do it here.

3) **Path**: The diameter of an n-node path is $n-1$ (see for example the 5-node path of Figure 6(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 connected to one node outside the circle; see example in Figure 8. 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, $\lim_{n\to\infty} H = 0$.

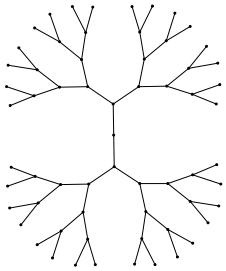


Fig. 7. An undirected force-model drawing of a binary tree.

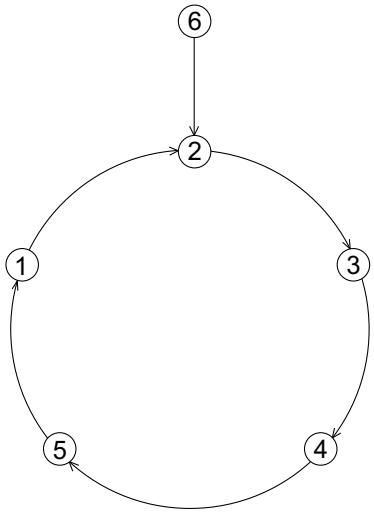


Fig. 8. A 5-node circle with an outlier.

D. Cyclic Digraphs

Standard layering algorithms can be applied only to acyclic digraphs. When dealing with cyclic digraphs they are first transformed into acyclic ones by inverting the direction of a minimal number of edges [3], [13]. (Although, in cases where a special root node is known in advance, better strategies are possible.) Our algorithm allows one to directly draw cyclic digraphs without having to invert edge directions. We believe this to be one of its most significant advantages.

To make this claim stronger, we now 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. As an example, in a directed cycle, like the one plotted in Figure 6(a), it suffices to invert the direction of a single edge in order to make it acyclic. Thus, the graph 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 9. Obviously, this misrepresents the structure of the hierarchy-free cycle. Applying our algorithm to a directed cycle 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 laid out in the same direction.

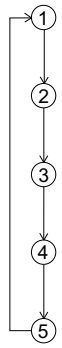


Fig. 9. Schematic layering of 5-points circle.

Another example is shown in Fig. 11(a). Here, the digraph does contain hierarchy, and the figure shows its optimal arrangement as dictated by our algorithm. We believe that we can quite objectively claim that this drawing best represents the structure of the digraph, despite of the fact that only about half of the edges point downward, and the rest point upward. 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. None of the other nodes possess evident hierarchical relations, thus some of the edges connecting them are 'allowed' to go upward.

VI. Examples

We have tested our algorithm against several unweighted digraphs with diverse structures. Most of the digraphs are based on matrices from the *Matrix Market* collection [23]. Each digraph is constructed by taking a matrix and replacing each non-zero entry (i,j) with a directed edge from i to j.

Figure 10 compares between the three different techniques for calculating the x-coordinates, with regard to three different digraphs. **Complete 5-level binary tree**: the optimal arrangement is naturally quantized into 6 layers, as dictated by the tree structure. Using the Fiedler vector to determine the x-coordinates obviously fails to capture the tree structure, with many nodes being placed in exactly the same location. This phenomenon is explained in Subsection III-B. Using the minimum linear arrangement we definitely reveal the tree structure, but the stress minimization technique unquestionably gives the best result. **Nos6 digraph** [23]: the three techniques all exhibit the symmetries of the graph very well. Notice the smooth layout formed by the Fiedler vector, as opposed to the more rigid one formed by the two other alternatives. **Nos3 digraph** [23]: Here, the superiority of the stress minimization technique is clear, with the Fiedler vector and minimum linear arrangement produce a condensed diagonal stripe that conceals the structure of this digraph.

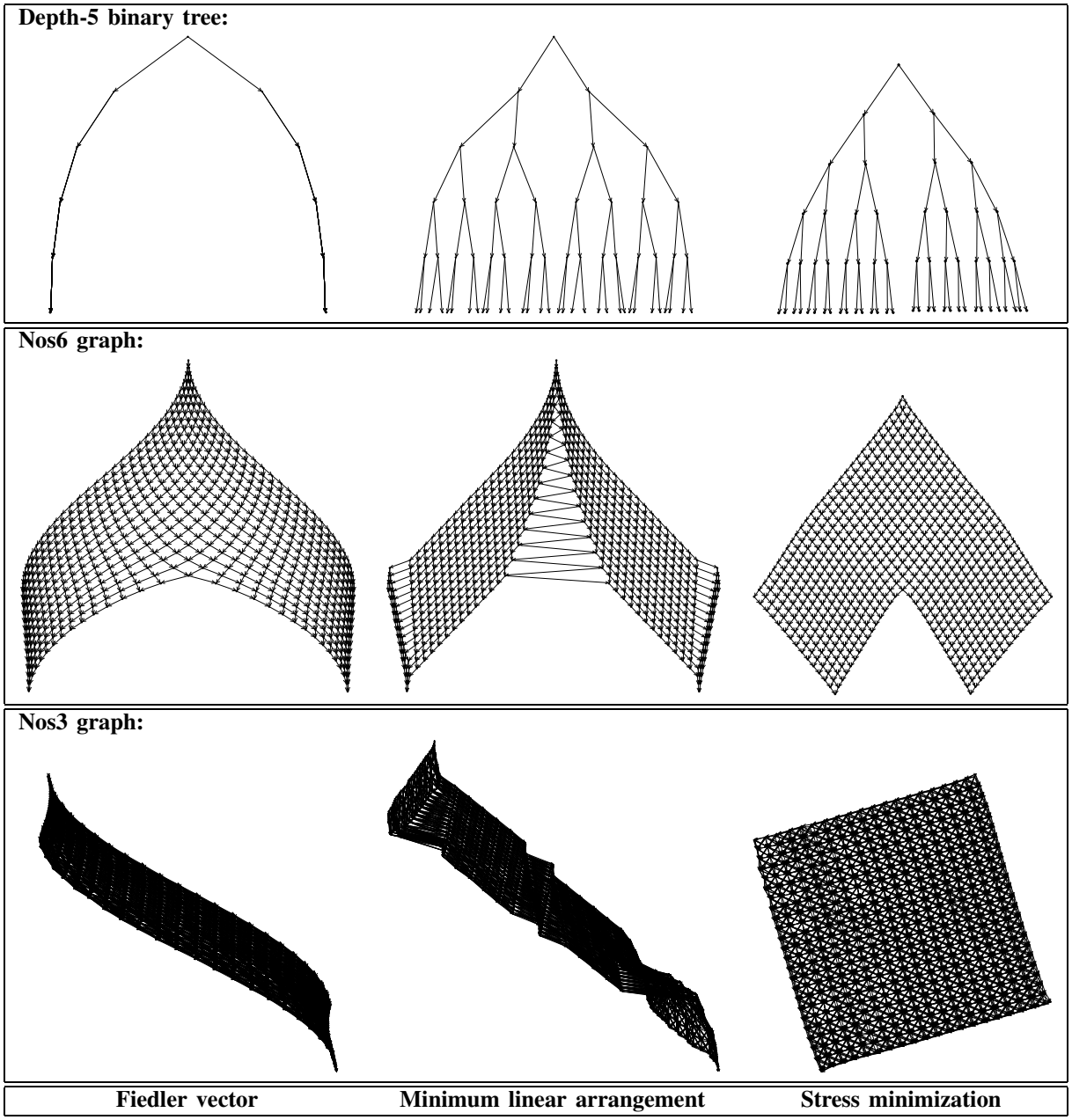


Fig. 10. Demonstration of drawings produced by our algorithm. The different columns are distinguished by the method used to produce the x-coordinates. The Fiedler vector was used in the left column, the minimum linear arrangement was used in the middle column, and the stress minimization was used in the right column. In the upper row a 5-node complete binary tree is shown. In the middle and lower rows the Nos6 and Nos 3 digraphs [23] are drawn. Obviously, the y-coordinates are identical for three versions of each digraph.

Figure 11 shows another three instructive examples. Here, we have used the Fiedler vector for drawing the x-coordinates, but the other two alternatives yield pretty much the same layouts. Figure 11(a) shows a directed cycle with an additional edge. In contrast to a pure cycle, which is regular and thus hierarchy-free, this digraph, thanks to the extra edge, does contain hierarchical information. Figure 11(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 11(c) is a cyclic version of the former digraph, with the direction of the edges along one of the paths (the middle one) 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. It seems that restricting the y-coordinates to strict horizontal layers would ruin the natural structure of the graphs of Fig. 11(b,c).

Figure 12 shows two different layouts of the Nos7 digraph [23]. In both cases the x-coordinates were constructed using the Fiedler vector. Here, the multiplicity of the lowest positive eigenvalue of the Laplacian is greater than 1, so there are two different Fiedler vectors. The left-hand-side drawing draws the graph in a "layering style", putting the nodes on many horizontal layers. The right-hand-side drawing has a three-dimensional look. It arranges the nodes in 9 two-dimensional

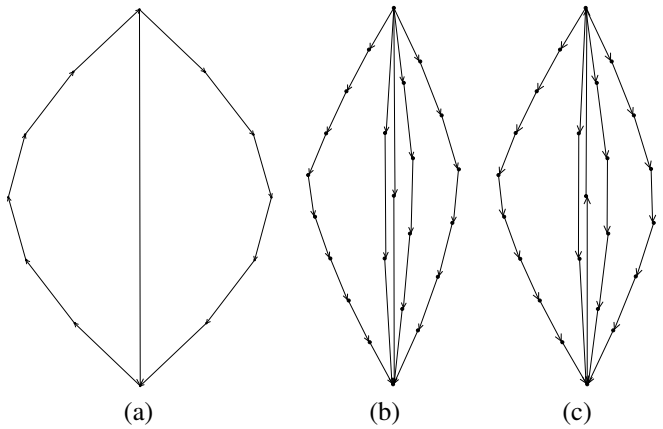


Fig. 11. Three examples of digraphs; **(a)** a distorted cycle. The extra edge is responsible for the observed hierarchy; **(b)** an acyclic digraph comprised of a few parallel paths with varying lengths; **(c)** a cyclic version of the former digraph.

layers. Note that in both drawings the edges point downwards.

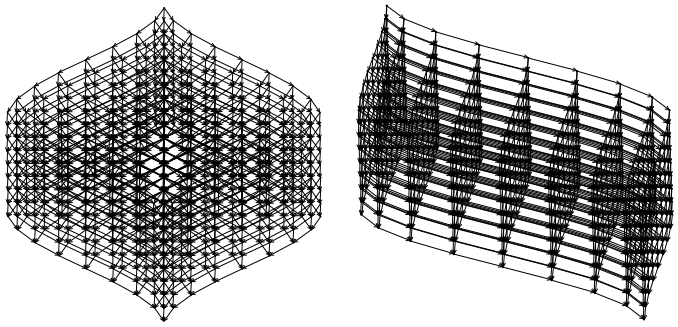


Fig. 12. Two different drawings of the Nos7 digraph [23]. The y-coordinates are the same, whereas the x-coordinates obtained by two different Fiedler vectors.

Frequently, drawing large digraphs is based on different aesthetic criteria than those used for drawing small digraphs. Nice drawings of small digraphs are usually obtained if one optimizes low-level features like minimizing the crossings between edges and/or node. This is, of course, very important for close examination of portions of the digraph. However, for large digraphs it might be better to focus on different, more global, aesthetic aspects, like expressing symmetries, in order to preserve the high-level structure. This allows the user to grasp important characteristics of the digraph simply by its visual inspection. In this regards, our method is more appropriate for the drawing of large digraphs, compared to the Sugiyama-style approach. To demonstrate this, we took two large digraphs and displayed them using both our algorithm and the *dot* software package [7], which is a state-of-the-art Sugiyama-style digraph drawing software; see Figure 13. The dot software puts much emphasize on the aforementioned low-level features like preventing overlap of nodes and optimizing the navigation of the edges. The general layout of Plat362 in Figure 13 is similar for both algorithms, but it is clear that our algorithm is somewhat superior in emphasizing the internal symmetries. The dot software seems to have more difficulty in capturing the structure of the Nos5 digraph, while it is clearly achieved using our algorithm. Another advantage of our algorithm in the context of drawing large digraphs, is in its computation speed; see Table I. The dot software may be many times slower.

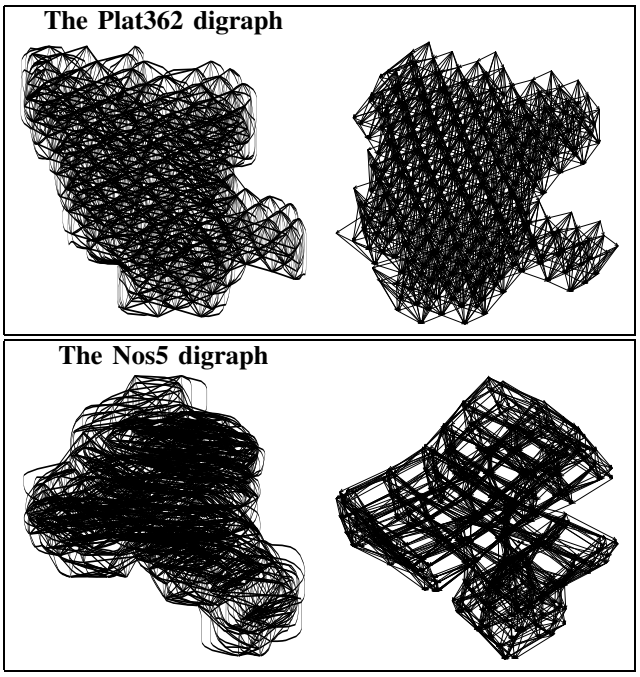


Fig. 13. Two large digraphs, Nos5 and Plat362 [23], were chosen for the comparison between our algorithm (the drawings on the right), and the popular dot software package (the drawings on the left). We have used stress minimization to determine the x-coordinates.

Another comparison between our algorithm and dot is presented in Figure 14, where the transcriptional gene regulation network of *Escherichia coli* is shown. Here, the two drawings are quite comparable, both capturing the structure of the network.

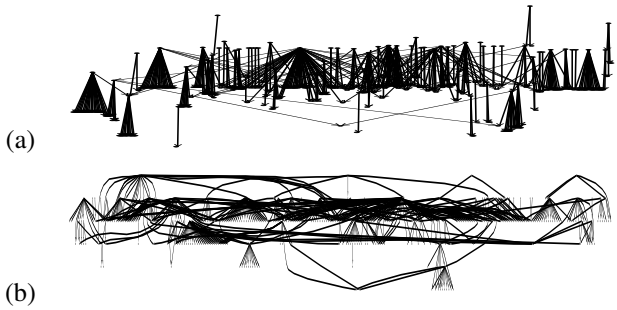


Fig. 14. Network of transcription interactions between regulatory proteins and genes in the bacterium *Escherichia coli*, taken from [19]. This digraph has $|V| = 424$ nodes and $|E| = 519$ edges. **(a)** The layout produced by our algorithm, where we have used minimum linear arrangement to find the x-coordinates. **(b)** The layout produced by the dot software.

Interestingly, our algorithm can be applied to graphs containing both directed and undirected edges. As was already mentioned, all we have to do in order to deal with an undirected edge (i, j) is to set $\delta_{ij} = \delta_{ji} = 0$, meaning that such an edge induces no hierarchical relation. Many graphs based on matrices in the Matrix Market collection [23]

contain both directed edges (when entry (i, j) is non-zero and entry (j, i) is zero) and undirected edges (when both entries (i, j) and (j, i) are non-zero). In Fig. 15 we show three such graphs: Dwa512, Dw2048 and Dw8192, all computed using the Fiedler vector for the x-coordinates. Directed edges are colored red and undirected edges are colored blue. In all the drawings the graph structure is shown nicely with excellent symmetry preservation. Moreover, all the directed edges point downwards, and induce hierarchical relations between nodes that are contained in undirected components of the graph. We think that these results demonstrate that sometimes restricting the nodes to strict layers hides the graph's natural structure.

In Table I we provide the sizes of the graphs and running times, as measured on a Pentium IV 2GHz PC with 256 MB RAM. In the table, the difference in computation speed between the three alternatives for the x-coordinates computation is apparent. Finding the Fiedler vector is unquestionably the fastest method, followed by the stress energy minimization, which is about two orders of magnitude slower, and then the minimum linear arrangement which is yet another order of magnitude slower. To get a feeling of the overall computation time, notice that the largest graph, containing 8192 nodes, was drawn in less than half a second using the Fiedler vector, while it required about half a minute using the minimum linear arrangement.

VII. Discussion

We have presented a digraph drawing algorithm that uses several one-dimensional energy minimization problems to find an optimal drawing in two-dimensions. The vector of y-coordinates is found using a rather elegant minimization of the so-called hierarchy energy, which yields a unique global minimizer that nicely captures the hierarchical information. To calculate the vector of x-coordinates, which contains non-directional information, we have proposed three possible energy functions. One of them, the stress energy function, can be minimized by an iterative process that is intimately related to the hierarchy energy. Thus, the hierarchy energy seems to be a fundamental concept in digraph drawing, involved with calculating the y, as well as the x, coordinates.

The layouts produced by our algorithm are very natural, and are not subject to any predefined restrictions. In a way, they simply "let the graph speak for itself". The fact that the layout is a minimizer of carefully designed energy functions enables a rather accurate representation of many properties of the graph, such as its hierarchical structure and its symmetries. In terms of running time, our algorithm is very fast, being able to draw 10,000-node graphs in less than a second on a mid-range PC.

Significant virtues of our algorithm include its ability to draw cyclic digraphs without having to invert edge directions, the possibility of applying it to graphs containing both directed and undirected edges and its ability to measure the amount of hierarchy in the digraph via the hierarchy index. One potential application of this index is to decide whether to use hierarchical drawing tools to represent a given digraph, or to prefer undirected graph drawing algorithms.

We believe this last issue to be worthy of further research, and suggest the possibility of combining digraph drawing algorithms and undirected graph drawing algorithms into a unified tool: Given a general digraph, we could 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 y-coordinates vector to find connected subgraphs, such that the nodes in each subgraph have similar y-coordinates. Such subgraphs are candidates for being hierarchy-free components, and should be processed separately.

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 have successive y-coordinates and no edge resides within a single subset. This might be specially useful for computing layering of cyclic digraphs, where other methods confront difficulties.
- **It can induce ordering**: Standard ordering algorithms are typically 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 our vector of x-coordinates to impose a "global ordering".

References

[1] U. Brandes, G. Shubina, R. Tamassia and D. Wagner, "Fast Layout Methods for Timetable Graphs", *Proc. Graph Drawing (GD'00)*, LNCS 1984, pp. 127–138, Springer-Verlag, 2000.

[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, *Graph Drawing: Algorithms for the Visualization of Graphs*, Prentice-Hall, 1999.

[4] J. Diaz, J. Petit and M. Serna, "A Survey on Graph Layout Problems", *ACM Computing Surveys* **34** (2002), 313–356.

[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] E. Gansner, E. Koutsofios and S. North, *Drawing Graphs with* dot, AT&T Labs — Research. Available at: `http://www.research.att.com/sw/tools/graphviz`

[8] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, USA, 1996.

[9] K. M. Hall, "An r-dimensional Quadratic Placement Algorithm", *Management Science* **17** (1970), 219–229.

[10] M. Jünger and P. Mutzel, "Exact and Heuristic Algorithms for 2-Layer Straightline Crossing Minimization", *Proc. Graph Drawing (GD'95)*, LNCS 1027, pp. 337–348, Springer-Verlag, 1995.

[11] T. Kamada and S. Kawai, "An Algorithm for Drawing General Undirected Graphs", *Information Processing Letters* **31** (1989), 7-15.

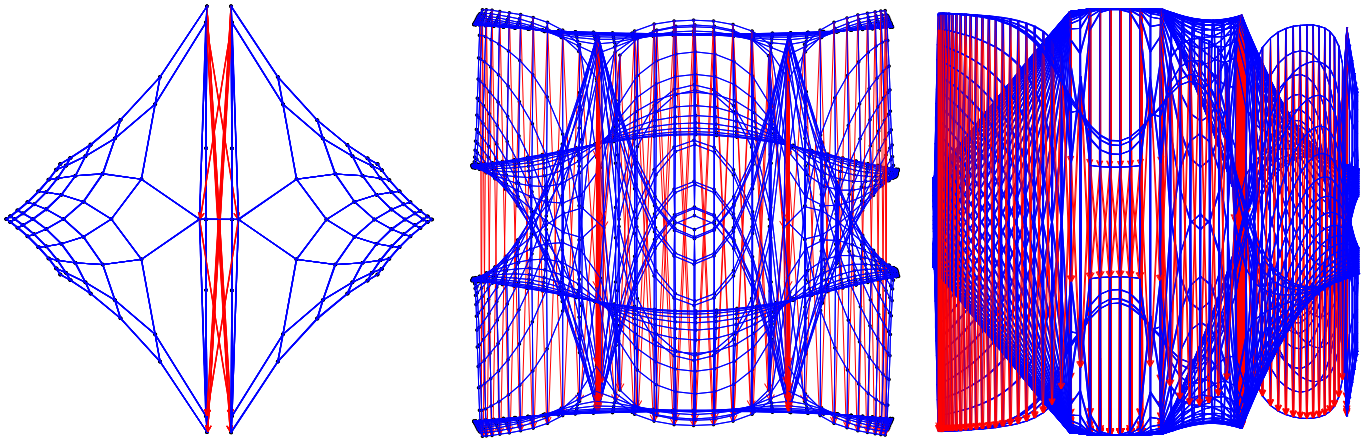
[12] T. Kamps, J. Kleinz and J. Read, "Constraint-Based Spring-Model Algorithm for Graph Layout", *Proc. Graph Drawing (GD'95)*, LNCS 1027, pp. 349–360, Springer-Verlag, 1995.

[13] M. Kaufmann and D. Wagner (Eds.), *Drawing Graphs: Methods and Models*, LNCS 2025, Springer Verlag, 2001.

[14] Y. Koren, "On Spectral Graph Drawing", *Proc. International Computing and Combinatorics Conference (COCOON'03)*, LNCS 2697, Springer-Verlag, 2003, to appear.

[15] Y. Koren, "One Dimensional Graph Drawing: Part II — Axis-by-Axis Stress Minimization", to be published.

[16] Y. Koren, L. Carmel and D. Harel, "ACE: A Fast Multiscale Eigenvectors Computation for Drawing Huge Graphs", *Proc. IEEE Symp. on Information Visualization 2002 (InfoVis 2002)*, IEEE computer society press, pp. 137–144 (2002).



The Dwa512 graph The Dw2048 graph The Dw8192 graph

Fig. 15. Graphs [23] containing both directed and undirected edges. Directed edges are colored by red, while undirected ones are colored by blue.

graph	|V|	|E|	y-coords time	x-coords time (E_{TH})	x-coords time (E_{LA})	x-coords time (E_S)
Nos3	960	7442	0.000	0.046	4.7	1
Nos5	468	2352	0.000	0.016	2.3	0.47
Nos6	675	1290	0.000	0.015	1.9	0.17
Nos7	729	1944	0.000	0.016	2.6	0.3
Dwa512	512	1004	0.000	0.016	1.8	0.41
Dw2048	2048	4094	0.015	0.11	6.7	1.9
Dw8192	8192	17,404	0.150	0.250	26	10
Plat362	362	2712	0.000	0.015	2	0.13

TABLE I

RUNNING TIME (IN SECONDS) OF THE ALGORITHM

[17] Y. Koren, L. Carmel and D. Harel, "Drawing Huge Graphs by Algebraic Multigrid Optimization", *Multiscale Modeling and Simulation*, SIAM, to appear.

[18] Y. Koren and D. Harel "A Multi-Scale Algorithm for the Linear Arrangement Problem", *Proc. Graph Theoretical Concepts in Computer Science 2002 (WG'02)*, LNCS 2573, Springer-Verlag, pp. 293–306, 2002.

[19] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii and U. Alon, "Network Motifs: Simple Building Blocks of Complex Networks", *Science* **298** (2002), 824–827.

[20] K. Sugiyama and K. Misue, "A Simple and Unified Method for Drawing Graphs: Magnetic-Spring Algorithm", *Proc. Graph Drawing (GD'94)*, LNCS 894, pp. 364–375, Springer-Verlag, 1995.

[21] K. Sugiyama, S. Tagawa and M. Toda, "Methods for Visual Understanding of Hierarchical Systems", *IEEE Transactions on Systems, Man, and Cybernetics* **11** (1981), 109–125.

[22] W. T. Tutte, "How to draw a graph", *Proceedings London Mathematical Society* **13** (1963), 743-768.

[23] The Matrix Market collection `math.nist.gov/MatrixMarket`

Liran Carmel received his BSc in physics from Tel-Aviv University, Israel, in 1991, and his MSc degree in physics from the Technion — Israel Institute of Technology, in 1998. He is currently completing his PhD studies in the Department of Computer Science and Applied Mathematics at the Weizmann Institute of Science, Israel. His research deals with materializing odor digitization, transmission and reproduction, and it involves the development of many kind of data visualization and classification algorithms. He is also interested in medieval pedigree visualization.

David Harel is the William Sussman Professor of Mathematics at The Weizmann Institute of Science in Israel, and has been Dean of the Faculty of Mathematics and Computer Science there since 1998. He is also co-founder of I-Logix, Inc., Andover, MA. He received his BSc from Bar-Ilan University in 1974, his MSc from Tel-Aviv University in 1976, and his PhD from MIT in 1978. He has worked in several areas of theoretical computer science, including computability, finite model theory and logics of programs, and in recent years has become involved in other areas, including software and systems engineering, visual languages, graph layout, modeling and analysis of biological systems, and smell communication. He is the inventor of statecharts, and co-inventor of live sequence charts (LSCs), and was part of the team that designed the Statemate and Rhapsody tools, and more recently the Play-Engine. He has received a number of awards, including ACM's Karlstrom Outstanding Educator Award in 1992. His latest books are "Dynamic Logic" (with Kozen and Tiuryn), MIT Press, 2000, "Computers Ltd.: What They Really Can't Do", Oxford, 2000, and "Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine", Springer-Verlag, 2003.

Yehuda Koren received his BA degree in computer science from the Open University, Israel, in 1997, and his MSc degree from the Weizmann Institute of Science, Israel, in 1999. Currently, he is completing his PhD studies in the Department of Computer Science and Applied Mathematics of the Weizmann Institute of Science. Among his primary research interests are algorithms for drawing large graphs, data analysis and visualization, multiscale optimization, and clustering algorithms.