Drawing Directed Graphs Using One-Dimensional Optimization ⋆

Liran Carmel, David Harel, and Yehuda Koren

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

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.

1 Introduction

Visualizing digraphs is a challenging task, requiring algorithms that faithfully represent the relative similarities of the nodes, and 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.* [11], 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 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. To make this feasible, *dummy nodes* are sometimes inserted. 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.

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, so as to minimize 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 [2,6].

⋆ For a full version see [1]

M.T. Goodrich and S.G. Kobourov (Eds.): GD 2002, LNCS 2528, pp. 193–206, 2002.

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. 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.

We define the vector of y-coordinates as the unique minimizer of a simple energy function, and show that the minimization problem is equivalent to a system of linear equations. The simple form of the energy function 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. As to the x-coordinates, they are assigned using the minimizer of another energy function that is suitable for the one-dimensional case.

By definition, a force is the inverse 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 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. Force directed models are much more popular in undirected graph drawing than in digraph drawing. We are aware of only one other occasion where a force directed model was suggested for digraph drawing, [10], 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 rather difficult to minimize.

2 The Algorithm

A digraph is usually written $G(V, E)$, 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. Each edge is associated with two magnitudes: **(1)** symmetric weights, $w_{ij} = w_{ji}$, and

(2) target height differences, which express the relative hierarchy of nodes i and j by the number δ_{ij}, measuring their desired height difference along the y-axis. Thus, in the drawing we would like to place nodes i and j such that $y_i - y_j = \delta_{ij} = -\delta_{ji}$. A digraph with $\delta_{ij} = 0$ for every $(i,j) \in E$ is really an undirected graph. A digraph is called *unweighted* if for every edge (i,j), $\delta_{ij} = w_{ij} = 1$. Henceforth, we assume $w_{ij} = 0$ for any non-adjacent pair of nodes.

By the conventional definition, the *Laplacian* is the symmetric $n \times n$ matrix

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

For later use, we introduce another important magnitude associated with a digraph:

Definition 1. *Let $G(V,E)$ be a digraph. The* **balance** *of node i is*

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

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 i measures the difference between how much i pushes away other nodes (those nodes j for which $\delta_{ij} > 0$), and how much it is pushed away from them.

2.1 Assigning the y-Coordinates

Here is our energy function:

Definition 2. *Let $G(V,E)$ 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 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.

Using the previously defined notions of Laplacian and balance, the hierarchy energy can be written in the compact form $E_H = E_0 + y^T L y - 2y^T b$, where $E_0 = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}\delta_{ij}^2$. For a proof see the full version of this paper [1]. Differentiating this simple form, 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 $G(V,E)$ be a digraph, with Laplacian L and balance b. An optimal arrangement $y^\star$ is a solution of $Ly = b$.*

The proof appears in [1], where we also show that the system $Ly = b$ is compatible with an infinite number of solutions that differ from each other only by a translation. The uniqueness (up to a translation) suggests that $y^\star$ carries some essential information, as we show in Section 3.

We are now in a position 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 coordinates, i.e., $\sum_i y_i^\star = 0$. This choice of $y^\star$ enables its fast computation using the *Conjugate Gradient method*, see [1]. Therefore:

Definition 3. *Let $G(V, E)$ be a digraph with Laplacian L and balance b. Its* **optimal arrangement***, $y^\star$, is the solution of $Ly = b$, subject to the constraint $\sum_i y_i = 0$.*

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

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

under the constraint $\sum_i y_i^\star = 0$. This is just the expected two-layer solution. The height difference between the layers is 1, thus $\delta_{ij} = y_i - y_j, \forall (i, j) \in E$, giving $E_H(y^\star) = 0$. Figure 1(b) shows another example of an unweighted acyclic digraph. In this case,

$$\begin{pmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix} y^\star = \begin{pmatrix} 2 \\ 0 \\ -2 \end{pmatrix} \qquad \Longrightarrow \qquad y^\star = \begin{pmatrix} 2/3 \\ 0 \\ -2/3 \end{pmatrix}.$$

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 $E_H(y^\star) = (\frac{2}{3} - 1)^2 + (\frac{2}{3} - 1)^2 + (\frac{4}{3} - 1)^2 = \frac{1}{3}$.

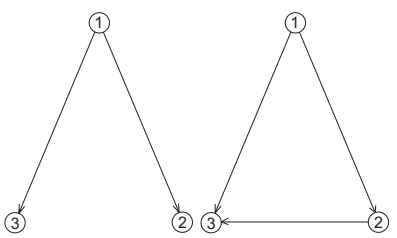


Fig. 1. Two very small examples of unweighted acyclic digraphs.

Figure 2(a) shows an example of an unweighted cyclic digraph. In this case,

$$\begin{pmatrix} 2 & -1 & 0 & -1 \\ -1 & 3 & -1 & -1 \\ 0 & -1 & 2 & -1 \\ -1 & -1 & -1 & 3 \end{pmatrix} y^\star = \begin{pmatrix} 0 \\ 1 \\ 0 \\ -1 \end{pmatrix} \quad \Longrightarrow \quad y^\star = \begin{pmatrix} 0 \\ 1/4 \\ 0 \\ -1/4 \end{pmatrix},$$

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, $E_H(y^\star) = 4 \cdot (\frac{1}{4} - 1)^2 + (\frac{1}{2} - 1)^2 = 2.5$.

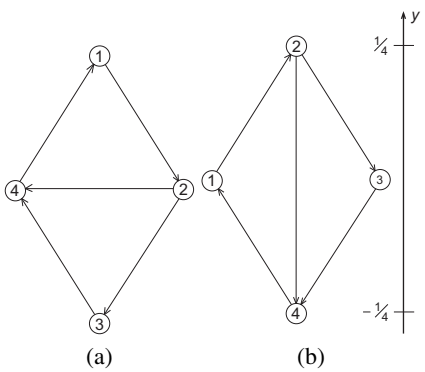


Fig. 2. (a) A small example of an unweighted cyclic digraph; **(b)** its optimal arrangement.

The idea of using energy minimization to determine a vector of coordinates on one axis was already exploited in undirected graph drawing by Tutte [12] and Hall [5], both utilizing the same quadratic energy function, $E_{TH} = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(y_i - y_j)^2 = y^T L y$. Comparing this energy with the hierarchy energy (1), it is clear that they become identical for a digraph with $\delta_{ij} = 0,\ \forall (i,j) \in E$, which is really an undirected graph.

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

$$y_i = \frac{\sum_{j=1}^{n} w_{ij} y_j}{\sum_{j=1}^{n} w_{ij}}. \tag{2}$$

This equation tells us to put node i in the *barycenter* of its neighbors. Clearly, the zero vector is a solution of (2), a situation that both Tutte and Hall avoid by using various constraints. In analogy, $y^\star$ satisfies the following important property

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

which is substantially different from (2). 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.

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'. 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"; see [1] for a demonstration of this.

It would be the best, then, to employ an alternative optimization technique for our one-dimensional case, which skirts the swapping problem We suggest to find a vector $x = (x_1, \ldots, x_n)^T$ representing the x-coordinates of the nodes in a way that minimizes edge (squared) lengths. This enables us to use simple energy functions that can be minimized by powerful global techniques. Besides the aesthetical reasoning, minimizing edge lengths is known to reduce edge crossings. We present two alternative variants of the method:

- **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 suggested by Hall [5,7], 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 [8] — an extremely fast multiscale algorithm for undirected graph drawing.
- **Minimizing edge lengths**: This is the well known problem of *minimum linear arrangement* [3]. 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 $(1, \ldots, n)$. 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 [9], designed especially for this problem.

In some of the cases we have studied (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. In contrast, the minimum linear arrangement incorporates some kind of 'repulsive force', by not letting two nodes have the same x-coordinate. For example, see Fig. 4(a,b). However the Fiedler vector has advantages that make it very attractive in some cases. First, it is guaranteed to be an optimal global minimizer, and second, it can be computed extremely rapidly; see Table 1. In practice, for many graphs the Fiedler vector yields better results (see Section 4).

Final beautification. So far, we have found the vector x using a very simple energy function and have ignored the information about the y-coordinates. As a result, there are cases where local smoothing can improve the drawing; see for example Fig. 4(c). We do this using a richer one-dimensional force directed model, which incorporates the information about the y-coordinates in order to refine the x-coordinates. Specifically, we have chosen to work with the Fruchterman-Reingold model [4]. In practice we do not use forces, but rather the energy they induce. This results in a modified optimization process, more suitable for the one-dimensional case; see [1].

3 Further Implications of the y-Axis Arrangement

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 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. It can be proved that in a regular digraph each node is balanced, having equal in-degree and out-degree. The optimal arrangement for a regular digraph is the solution of $Ly^\star = 0$; hence $y^\star = (0, \ldots, 0)^T$. In other words, a regular digraph is a hierarchy-free digraph. A simple example is a directed cycle.

Symmetric Nodes A regular digraph is hierarchy-free. Thus, when the nodes are all symmetric, they are all assigned the same y-coordinate. Interestingly, this observation can be extended. Actually, 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 by adding the requirement that $b = b^\pi$. 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, which is indeed the case. In [1], using the uniqueness of the optimal arrangement, we prove that if i and j are symmetric, then $y_i^\star = y_j^\star$.

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: **(1)** A directed cycle, in which no node is different from the other, and we do not expect to see any hierarchy at all. Indeed, it is the case, since the cycle is regular. **(2)** A $(k+1)$-path has a maximal amount of hierarchy, each node having a different y-coordinate in unit increments, $y^\star = (-k/2, -k/2+1, \ldots, k/2)^T$.

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 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 graph, 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 theoretic distance between the two most distant nodes. Therefore:

Definition 4. *The* **hierarchy index** *of an unweighted digraph 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. Regular digraphs are an extreme case of the latter scenario. Here $\Delta y^\star = 0$, and hence $H = 0$. The dual example is a k-path, where $\Delta y^\star = D = k-1$, implying $H = 1$ independent of k. Another example is that of a complete n-node binary tree. Here, $D = 2\log n$, while $\Delta y^\star = \log n$. Hence, $H = \frac{1}{2}$, independently of n. This 1:2 ratio is well visualized when comparing the height of a hierarchical tree drawing with that of a radial drawing generated by undirected force models; see [1].

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 [2,6]. (Although, in cases where a special root node is known in advance, better strategies are possible.) Our algorithm allows 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 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 3. 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.

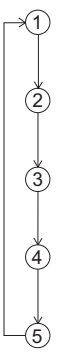


Fig. 3. Schematic layering of 5-points circle.

Another example is shown in Fig. 5(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.

4 Examples

We have tested our algorithm against several unweighted digraphs with diverse structures. Figure 4 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 can be done in two different ways and that there is also a possibility for an additional final beautification. Figure 4(a) shows the drawing obtained when using the Fiedler vector. The result is really bad, with many nodes being placed in exactly the same location. This phenomenon is explained in Subsection 2.2. Using the minimum linear arrangement we get a much better drawing, shown in Fig. 4(b). In Fig. 4(c) we show a slightly improved result produced by the final beautification.

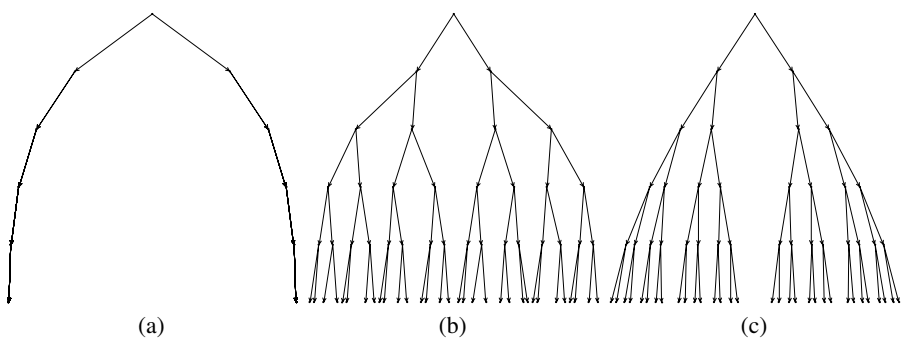


Fig. 4. A 5-level complete binary tree **(a)** x-coordinates obtained using the Fiedler vector; **(b)** x-coordinates obtained using the minimum linear arrangement; **(c)** applying final beautification to refine x-coordinates of (b)

Figure 5 shows three instructive examples. Figure 5(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 5(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 5(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

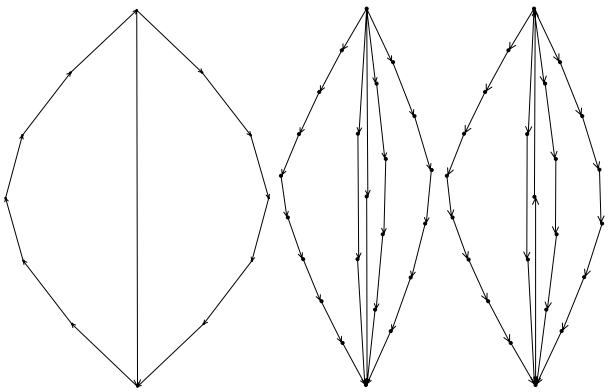


Fig. 5. Three examples of digraphs; **(a)** a distorted cycle. The extra edge is responsible for the observed hierarchy; **(b)** an acyclic digraph comprised of a few parallel paths with varying lengths; **(c)** a cyclic version of the former digraph.

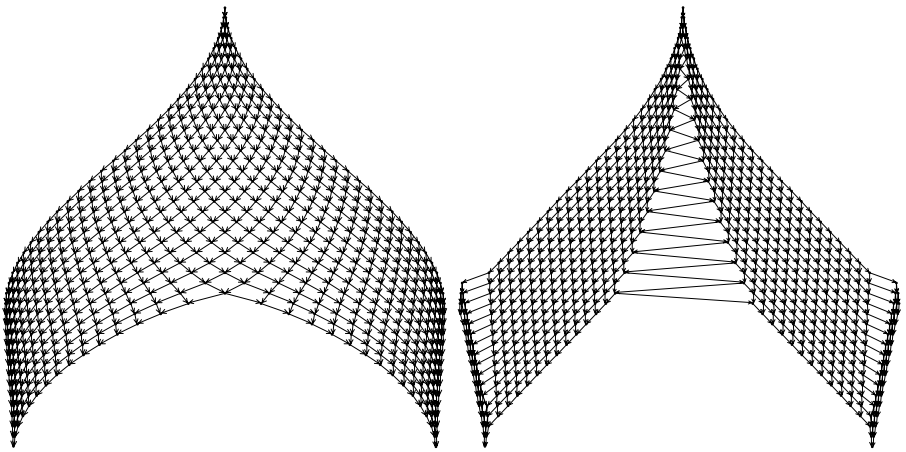
"reversed" edges. It seems that restricting the y-coordinates to strict horizontal layers would ruin the natural structure of the graphs of Fig. 5(b,c).

In the rest of this section, we present graphs which are based on matrices from the *Matrix Market* collection (available at: `math.nist.gov/MatrixMarket/`). Each graph is constructed by taking a matrix and replacing each non-zero entry (i, j) with a directed edge from i to j. In Table 1 we provide the sizes of the graphs and running times, as measured on a Pentium IV 2GHz PC. The largest graph, containing 8192 nodes, was drawn in less than $\frac{1}{2}$ second. Unless otherwise is stated, we computed the x-coordinates using the Fiedler vector (without final beautification).

Table 1. Running time (in seconds) of the the algorithm

graph	$\|\mathbf{V}\|$	$\|\mathbf{E}\|$	x**-coords time**	y**-coords time**	**total time**
Nos6	675	1290	0.000	0.015	0.015
Nos7	729	1944	0.000	0.016	0.016
Dwa512	512	1004	0.000	0.016	0.016
Dw2048	2048	4094	0.015	0.11	0.125
Dw8192	8192	17,404	0.150	0.250	0.4

Figure 6 shows two different drawings of the Nos6 graph. For both of them the y-coordinates are the same (as is always the case, since there is a unique minimizer of the hierarchy energy). However, the x-coordinates are different. In Fig. 6(a) they were computed using the Fiedler vector, while in Fig. 6(b) they were computed using the minimum linear arrangement. Both drawings exhibit the symmetries of the graph very well. Regarding running times, computation of the minimum linear arrangement took 3.7sec, whereas computation of the Fiedler vector took only 0.015sec.



(a) (b)

Fig. 6. The Nos6 graph. The x-coordinates are drawn by **(a)** Fiedler vector; **(b)** minimum linear arrangement.

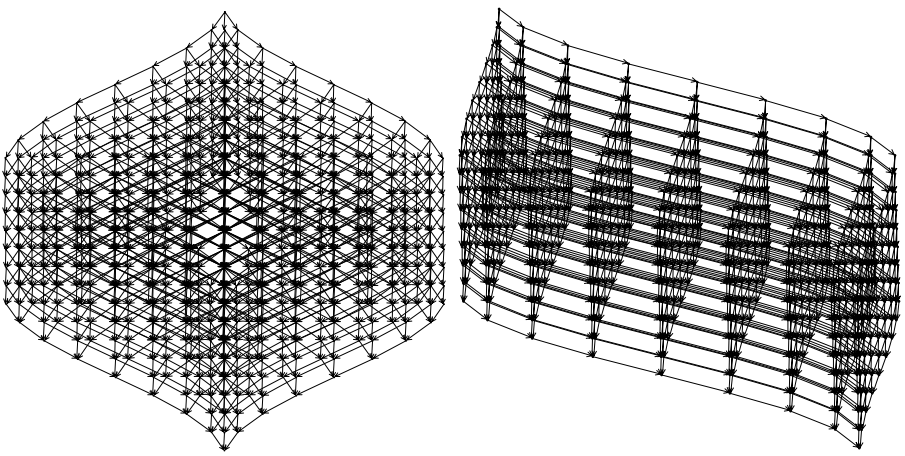
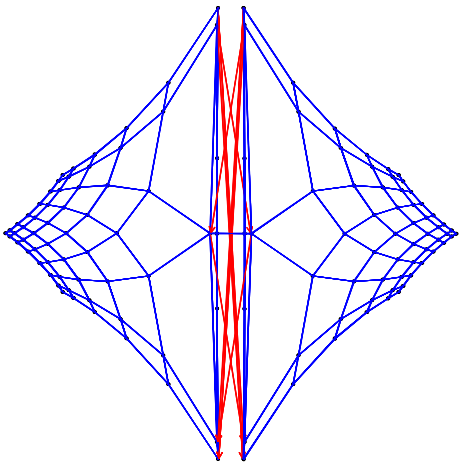
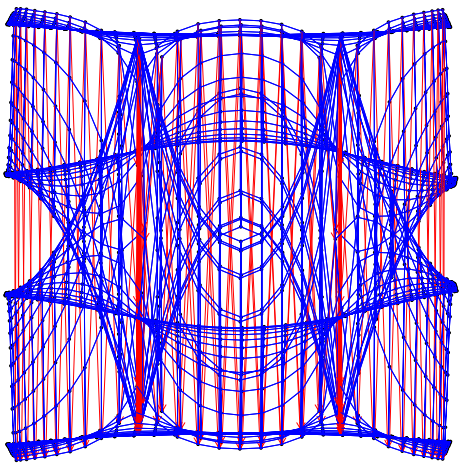


Fig. 7. Two different drawings of the Nos7 graph. The y-coordinates are the same, whereas the x-coordinates obtained by two different Fiedler vectors.

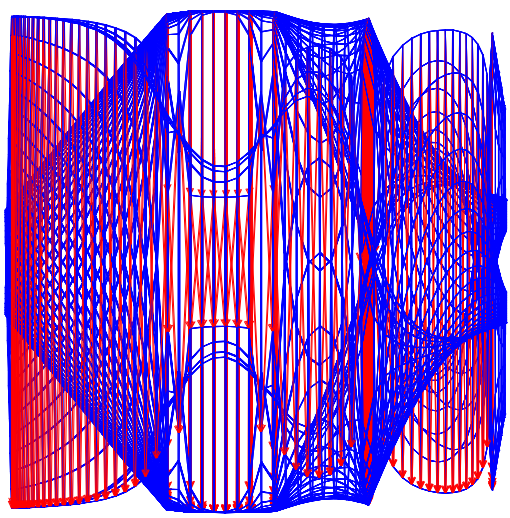
Figure 6 shows two different layouts of the Nos7 graph. In both cases the x-coordinates were constructed using the Fiedler vector. Here, the multiplicity of the lowest positive eigenvalue of the Laplacian matrix is greater than 1, so there are two different Fiedler vectors. The left-hand-side drawing 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 layers. Note that in both drawings the edges point downwards.



The Dwa512 graph



The Dw2048 graph



The Dw8192 graph

Fig. 8. Graphs containing both directed and undirected edges. Directed edges are colored by red, while undirected edges are colored by blue (see electronic version of this paper).

Interestingly, our algorithm can be applied to graphs containing both directed and undirected edges. As was already mentioned, all we have to do 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 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. 8 we show three such graphs: Dwa512, Dw2048 and Dw8192. 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. Note that all directed edges point downwards, and that they 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.

5 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 energy minimization algorithm, which yields a unique global minimizer that nicely captures the hierarchical information. For the vector of x-coordinates, which contains non-directional information, we are using an optimization algorithm especially tailored for the one dimensional problem.

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 global minimizer of the one-dimensional energies 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. The amount of hierarchy can be used 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.
- **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 the vector of x-coordinates obtained by our 'first stage' to impose a 'global ordering'.

References

1. L. Carmel, D. Harel and Y. Koren, "Drawing Directed Graphs Using One-Dimensional Optimization", Technical Report MCS02-14, The Weizmann Institute of Science, 2001. Available on the web.
2. G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*, Prentice-Hall, 1999.
3. J. Diaz, J. Petit and M. Serna, "A Survey on Graph Layout Problems", to appear in *ACM Computing Surveys*.
4. T. M. G. Fruchterman and E. Reingold, "Graph Drawing by Force-Directed Placement", *Software-Practice and Experience* **21** (1991), 1129-1164.
5. K. M. Hall, "An r-dimensional Quadratic Placement Algorithm", *Management Science* **17** (1970), 219-229.
6. M. Kaufmann and D. Wagner (Eds.), *Drawing Graphs: Methods and Models*, Lecture Notes in Computer Science, Vol. 2025, Springer Verlag, 2001.
7. Y. Koren, "On Spectral Graph Drawing", manuscript, 2002.
8. 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, to appear.
9. Y. Koren and D. Harel "A Multi-Scale Algorithm for the Linear Arrangement Problem", *Proc. Graph Theoretical Concepts in Computer Science 2002 (WG 2002)*, Springer-Verlag, to appear.
10. K. Sugiyama and K. Misue, "A Simple and Unified Method for Drawing Graphs: Magnetic-Spring Algorithm", *Proc. Graph Drawing 1994*, Lecture Notes in Computer Science, Vol. 894, pp. 364–375, Springer Verlag, 1995.
11. K. Sugiyama, S. Tagawa and M. Toda, "Methods for Visual Understanding of Hierarchical Systems", *IEEE Transactions on Systems, Man, and Cybernetics* **11(2)** (1981), 109–125.
12. W. T. Tutte, "How to draw a graph", *Proc. London Math. Society* **13** (1963), 743-768.