On Layering Directed Acyclic Graphs

Martin Harrigan*and Patrick Healy†
Department of Computer Science and Information Systems,
University of Limerick, Limerick, Ireland

Abstract

We consider the problem of layering a directed acyclic graph with minimum dummy nodes. We present a new Integer Linear Programming formulation of the problem based on a set of fundamental cycles in the underlying undirected graph and show that it can be solved in polynomial time. We outline some of the advantages of the formulation. Each solution defines a family of layerings with the same number of dummy nodes. We can also transform one solution into another by adding or removing certain combinations of dummy nodes, thus allowing the consideration of other aesthetics.

1 Introduction

Graph layering is an important step in the Sugiyama framework [5] for drawing directed acyclic graphs (DAGs). A graph layering algorithm partitions the node set of a graph into subsets, called layers, so that all the edges point in the same direction and edges only occur between consecutive layers. Whenever an edge is about to cross a layer, a dummy node is added. We would like a graph layering algorithm to minimize the number of dummy nodes for the following reasons [1, p.271]:

- A small number of dummy nodes generally results in compact drawings;
- The time required by subsequent steps of the Sugiyama framework [5] depends on the total number of nodes (real nodes plus dummy nodes);
- Bends in the final drawing occur only at dummy nodes and reduce readability.

Existing layering algorithms include the Longest Path algorithm, the Coffman-Graham algorithm [2] and Gansner et al.'s network simplex algorithm [3]. The

*Supported by the Irish Research Council for Science, Engineering and Technology: funded by the National Development Plan
†{martin.harrigan, patrick.healy}@ul.ie

Longest Path algorithm layers a graph with minimum height. The Coffman-Graham algorithm layers a graph with width at most W and height $H < (2 - 2/W)H_{min}$, where H_{min} is the minimum height of a layering of width W. The network simplex algorithm layers a graph with minimum dummy nodes, however, it has not been proven to have polynomial running time. Other approaches to layering a graph with minimum dummy nodes involve solving a linear program using interior-point methods or converting the program to an equivalent minimum-cost flow problem and using an algorithm like Orlin's [4]. While these methods are polynomial time they are difficult to implement in practice.

This paper is organised as follows. In the following section we begin with some definitions. In Section 3 we define a layering in two ways; the first defines the layer of each real node while the second defines explicitly the location of the dummy nodes. We also present the corresponding Integer Linear Programming (ILP) formulations. In Section 4 we outline some of the advantages of our new formulation. Finally, in Section 5 we draw some conclusions from our work.

2 Definitions

A *graph* $G = (V, E)$ consists of a *vertex* set V and an *edge* set E, where an edge is an unordered pair of distinct vertices of G. A *cycle* is a sequence of edges ($e_i \in E$)

$$\{e_0, e_1, \ldots, e_n\}$$

such that e_i and $e_{i+1} \pmod n$ share a vertex, any vertex is shared between at most two of the edges, and no edge appears more than once. An orientation of a graph G is a function σ from the edges of G to $\{-1, 1\}$ such that if (u, v) is an edge, then $\sigma(u, v) = -\sigma(v, u)$.

A *directed graph* $D = (V, A)$ consists of a vertex set V and an *arc* set A, where an arc, or *directed edge*, is an ordered pair of distinct vertices of D. A *directed cycle* is a sequence of arcs ($a_i \in A$)

$$\{a_0, a_1, \ldots, a_n\}$$

such that if $a_i = (u_i, v_i)$ then $v_i = u_{i+1} \pmod n$, and no arc appears more than once. A *directed acyclic graph* (DAG) is a directed graph that contains no directed cycles. Since any DAG has no symmetric pair of arcs, it can be represented by its underlying undirected graph together with an orientation (see Figure 1).

A *hypergraph* $H = (V, \mathcal{E})$ consists of a vertex set V and a *hyperedge* set $\mathcal{E}$, where a hyperedge is a non-empty subset of vertices of H.

2.1 Cycle Bases and Fundamental Cycles

Let $D = (V, A)$ be a DAG and $G = (V, E)$ be the underlying undirected graph of D with orientation σ. Although D is acyclic, G may contain cycles. Let $\mathcal{C}$ be

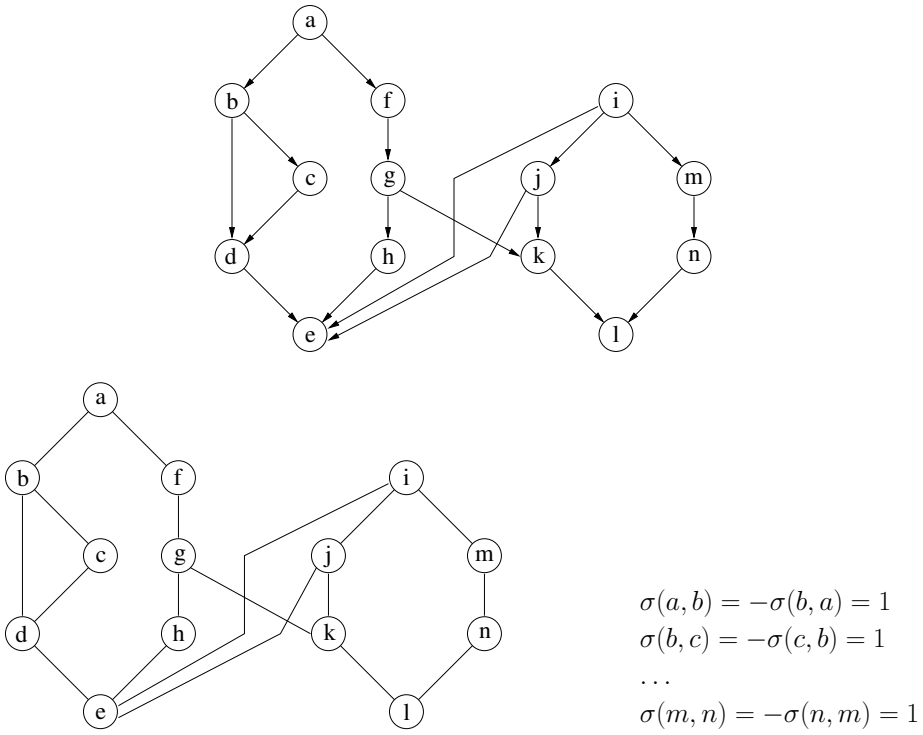


Figure 1: A DAG, its underlying undirected graph and orientation.

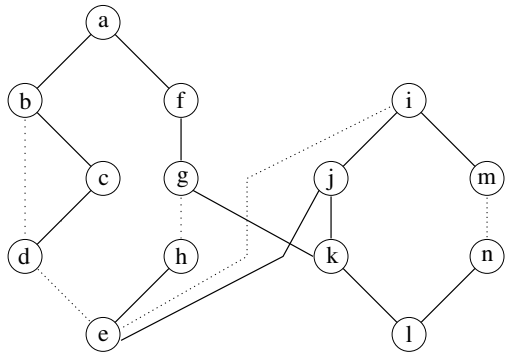


Figure 2: A spanning tree (full lines) and chords (dotted lines) for the undirected graph in Figure 1.

$\mathcal{F}$	Chords	Fundamental Cycles
F_0	(b,d)	$\{(b,d),(d,c),(c,b)\}$
F_1	(d,e)	$\{(d,e),(e,j),(j,k),(k,g),(g,f),(f,a),(a,b),(b,c),(c,d)\}$
F_2	(g,h)	$\{(g,h),(h,e),(e,j),(j,k),(k,g)\}$
F_3	(e,i)	$\{(i,e),(e,j),(j,i)\}$
F_4	(m,n)	$\{(m,n),(n,l),(l,k),(k,j),(j,i),(i,m)\}$

Table 1: Chords and fundamental cycles for the spanning tree in Figure 2.

the set of all cycles in G. The *cycle vector* χ_C of a cycle $C \in \mathcal{C}$ (with coordinates $\chi_C(e)$, $e \in E$) is defined by

$$\chi_C(e) = \begin{cases} 1 \text{ if } e \in C \\ 0 \text{ if } e \notin C. \end{cases}$$

The *cycle vector space* of G is the vector space over $GF(2)$ spanned by χ_C ($\forall C \in \mathcal{C}$). A set of cycles $\mathcal{B} = \{B_1, \ldots, B_n\} \subseteq \mathcal{C}$ is a *cycle basis* if χ_B ($\forall B \in \mathcal{B}$) form a basis for the cycle vector space of G. Therefore, for every $C \in \mathcal{C}$,

$$\chi_C = \sum_{i=1,\ldots,n} \alpha_i B_i$$

with at least one non-zero coefficient.

Let T be a spanning forest of G. A set of *fundamental cycles* $\mathcal{F}$ can be constructed as follows: For each $e = (u,v) \notin T$ there is a unique cycle in $T \cup \{e\}$ (see Figure 2 and Table 1). Each e is a *chord* of G with respect to T. A set of fundamental cycles constitutes a cycle basis, however, the converse is not necessarily true. The orientation of the chords provide a natural direction in which to traverse the fundamental cycles. *Forward* (resp., *backward*) edges in

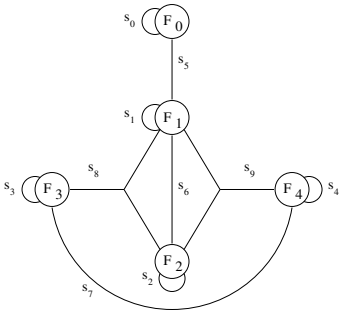


Figure 3: The fundamental cycle hypergraph of the undirected graph in Figure 1 with respect to the set of fundamental cycles in Table 1.

Hyperedge	Associated edge subset ($s_i = s_{i+} \cup s_{i-}$)	
s_0	$\{(b,d)\}$	$\emptyset$
s_1	$\{(d,e),(a,b)\}$	$\{(f,a),(g,f)\}$
s_2	$\{(g,h),(h,e)\}$	$\emptyset$
s_3	$\{(i,e)\}$	$\emptyset$
s_4	$\{(m,n),(n,l),(i,m)\}$	$\{(l,k)\}$
s_5	$\{(b,c),(c,d)\}$	$\emptyset$
s_6	$\{(g,k)\}$	$\emptyset$
s_7	$\emptyset$	$\{(j,i)\}$
s_8	$\emptyset$	$\{(e,j)\}$
s_9	$\{(j,k)\}$	$\emptyset$

Table 2: The edge subsets of the undirected graph in Figure 1 associated with each hyperedge of the hypergraph in Figure 3.

a fundamental cycle are those edges $(u, v) \in E$ that are oriented with (resp., against) this direction.

A *fundamental cycle hypergraph* of G with respect to $\mathcal{F}$ is the hypergraph $FCH_{(G,\mathcal{F})}(\mathcal{F}, \mathcal{E})$ where the vertex set is $\mathcal{F}$, and where each hyperedge $E \in \mathcal{E}$ is a maximal non-empty subset $E \subseteq \mathcal{F}$ such that $\bigcap_{F \in E} F \neq \emptyset$. Each hyperedge is therefore associated with a subset of G's edges (see Figure 3 and Table 2).

3 The Layering Problem

A *layering* of D is a partition of V into subsets $L_1, \ldots, L_h$, such that if $(u, v) \in A$, where $u \in L_i$ and $v \in L_j$, then $i > j$. The *span* of an edge (u, v) with $u \in L_i$ and $v \in L_j$ is $i - j$. The layering is *proper* if no edge has span greater than one. This can be achieved by adding *dummy nodes* along edges whose edge span is greater than one.

A natural way to define a layering is by a function that maps each node to an integer representing the layer,

$$\mathcal{L} : V \rightarrow \mathbb{Z}$$

and satisfies the following constraints:

$$\mathcal{L}(u) > 0, \quad \forall u \in V \tag{1}$$

$$\mathcal{L}(u) - \mathcal{L}(v) > 0, \quad \forall (u, v) \in A. \tag{2}$$

The location of the dummy nodes are implicity defined by $\mathcal{L}$; if $(u, v) \in A$ then there are $\mathcal{L}(u) - \mathcal{L}(v) - 1$ dummy nodes along the arc (u, v).

There is an alternative way to define a layering that explicitly locates the dummy nodes,

$$\mathcal{L}' : E \rightarrow \mathbb{Z}.$$

The domain for this function is the set of undirected edges in G. The difficulty now lies in determining what constraints $\mathcal{L}'$ must satisfy in order to represent a proper layering. We introduce the notion of a *balanced* cycle in the underlying undirected graph. $\mathcal{L}'$ balances a cycle C in G if

$$\sum_{e=(u,v) \in E} \chi_C(e) \sigma(u, v) \left(1 + \mathcal{L}'(e)\right) = 0,$$

where the edges $(u, v) \in E$ are traversed in an arbitrary direction around the cycle.

Proposition. *If $\mathcal{L}'$ balances any set of fundamental cycles in G, then it balances all the cycles in G.*

Proof. Let $\mathcal{F} = \{F_1, \ldots, F_m\}$ be any set of balanced fundamental cycles in G. Therefore,

$$\sum_{e=(u,v) \in E} \chi_{F_i}(e) \sigma(u, v) \left(1 + \mathcal{L}'(e)\right) = 0, \quad i = \{1, \ldots, m\},$$

Algorithm 1 PARTITION-EDGE-SET

Input: a graph $G = (V, E)$ and a set of fundamental cycles $\mathcal{F} = \{F_1, \ldots, F_m\}$
Output: a partition of the edge set $S = \{s_1, \ldots, s_n\}$

construct a *fundamental cycle - edge incidence* matrix of G as follows $\begin{bmatrix} \chi_{F_1} \\ \ldots \\ \chi_{F_m} \end{bmatrix}$

sort the columns using radix sort, where each column represents a binary number, thereby grouping identical columns together

partition the edge set into subsets of identical columns, $S = \{s_1, \ldots, s\}$

where the edges $e \in E$ are traversed in the direction identified by the orientation of the branches of the spanning tree used to construct $\mathcal{F}$.

Let C be any cycle in G. The incidence vector of C is some linear combination over $GF(2)$ of the cycles in $\mathcal{F}$, therefore,

$$\chi_C = \sum_{i=1}^{m} \lambda_i \chi_{F_i}$$

such that at least one λ_i is non-zero. So,

$$\begin{aligned} &\sum_{e=(u,v)\in E} \chi_C(e)\sigma(u,v)\,(1+\mathcal{L}'(e)) \\ &\qquad = \sum_{e=(u,v)\in E} \sum_{i=1}^{m} \lambda_i \chi_{F_i}(e)\sigma(u,v)\,(1+\mathcal{L}'(e)) \\ &\qquad = \lambda_i \sum_{i=1}^{m} \sum_{e=(u,v)\in E} \chi_{F_i}(e)\sigma(u,v)\,(1+\mathcal{L}'(e)) \\ &\qquad = \lambda_i \sum_{i=1}^{m} (0) = 0. \end{aligned}$$

□

Clearly, if all the cycles in G are balanced, then D is properly layered.

$\mathcal{L}'$ specifies the individual edges that are assigned dummy nodes. However, this can be generalised as follows. We partition the edge set with respect to some set of fundamental cycles $\mathcal{F}$ so that each subset is a maximal set of edges shared between the same subset of $\mathcal{F}$. These are the same subsets that are associated with the hyperedges of the fundamental cycle hypergraph $FCH_{(G,\mathcal{F})}$.

Let $S = \{s_1, \ldots, s_n\}$ be such a partition. It can be determined using PARTITION-EDGE-SET in $O(|E|^2)$ time. The subset of edges in $s_i \in S$ can be further divided into two groups, s_{i+} and s_{i-}, those pointing in one direction and

those point in the opposite direction. We again differentiate these as forward and backward edges. Any dummy node assigned to a forward (resp., backward) edge can be moved to any other forward (resp., backward) edge within the same edge subset while still ensuring that the fundamental cycles remain balanced.

We replace $\mathcal{L}'$ with two functions,

$$\begin{aligned}\mathcal{L}'_+ &: S \to \mathbb{Z} \text{ and}\\ \mathcal{L}'_- &: S \to \mathbb{Z}\end{aligned}$$

where $\mathcal{L}'_+(s)$ and $\mathcal{L}'_-(s)$ are the number of dummy nodes assigned to the subset of edges $s \in S$ on the forward and backward edges respectively. The functions are subject to the following constraints:

$$\sum_{s\in S} \gamma(F,s)\left(|s_+| - |s_-| + \mathcal{L}'_+(s) - \mathcal{L}'_-(s)\right) = 0, \quad \forall F \in \mathcal{F}, \tag{3}$$

$$\mathcal{L}'_+(s) = 0, \quad \forall s \in S \text{ such that } s_+ = \emptyset, \tag{4}$$

$$\mathcal{L}'_-(s) = 0, \quad \forall s \in S \text{ such that } s_- = \emptyset, \tag{5}$$

$$\mathcal{L}'_+(s) \geq 0, \mathcal{L}'_-(s) \geq 0, \quad \forall s \in S \tag{6}$$

where $\gamma(F,s)$ is the orientation of the edge subset s within the fundamental cycle F and is defined as follows: $\gamma(F,s) = 1$ if $s \subseteq F$ and the forward (resp. backward) edges in s are forward (resp. backward) edges in F, $\gamma(F,s) = -1$ if $s \subseteq F$ and the forward (resp. backward) edges in s are backward (resp. forward) edges in F, and $\gamma(F,s) = 0$ if $s \not\subseteq F$. The function $\gamma(s,F)$ defines a *fundamental cycle - edge subset incidence matrix* of D.

3.1 The ILP Formulations

Gansner et al. [3] formulated the layering problem as an ILP based on $\mathcal{L}$ where the objective function is:

$$\min \quad \textstyle\sum_{(u,v)\in A} \quad \mathcal{L}(u) - \mathcal{L}(v)$$

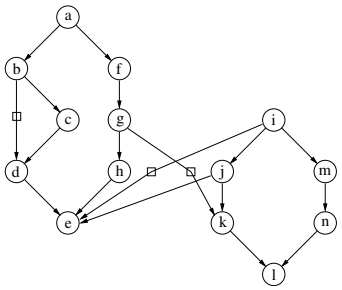
subject to (1), (2) and integrality constraints on all the variables.

An alternative ILP formulation based on $\mathcal{L}'_+$ and $\mathcal{L}'_-$ is:

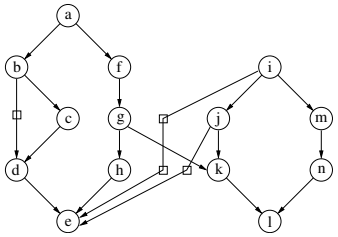
$$\min \quad \textstyle\sum_{s\in S} \quad \mathcal{L}'_+(s) + \mathcal{L}'_-(s)$$

subject to (3), (4), (5), (6) and integrality constraints on all the variables.

The constraint matrix of our formulation, like that of Gansner et al. [3], has a property known as *total unimodularity*. A matrix is totally unimodular if and only if every submatrix has determinant 0, 1, or -1. This ensures that the corresponding relaxed linear program has integral solutions and can be solved in polynomial time.



(a) Minimum Dummy Node Layering



(b) Longest Path Layering

Figure 4: Different layerings of the graph in Figure 1.

4 Advantages

4.1 A Family of Layerings

A solution in the form of functions $\mathcal{L}'_+$ and $\mathcal{L}'_-$ describes a family of layerings with the same number of dummy nodes. The forward and backward dummy nodes assigned to each edge subset can be distributed in any way amongst the forward and backward edges respectively within that edge subset. Since m indistinguishable dummy nodes can be distributed between n distinguishable edges in $\begin{pmatrix} m+n-1 \\ n-1 \end{pmatrix}$ distinct ways then a solution defines

$$\prod_{s \in S} \begin{pmatrix} \mathcal{L}'_+(s) + |s_+| - 1 \\ |s_+| - 1 \end{pmatrix} \begin{pmatrix} \mathcal{L}'_-(s) + |s_-| - 1 \\ |s_-| - 1 \end{pmatrix}$$

distinct layerings . From this family of layerings, we can choose one that satisfies other preferences. For example, if a user wanted a particular edge $e \in s$ to be as short as possible within the confines of a minimum dummy node solution, we could move the dummy nodes from e to other edges of the same direction within s.

4.2 Transforming Solutions

If we layer the graph in Figure 1 with the minimum number of dummy nodes (see Figure 4a) by solving the ILP above, the dummy nodes are assigned to the edge subsets s_0, s_3 and s_6 as follows:

$$\begin{aligned} \mathcal{L}'_+(s_0) &= 1, \\ \mathcal{L}'_+(s_3) &= 1, \text{ and} \\ \mathcal{L}'_+(s_6) &= 1. \end{aligned}$$

	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9
F_0	1	0	0	0	0	-1	0	0	0	0
F_1	0	1	0	0	0	1	-1	0	1	1
F_2	0	0	1	0	0	0	-1	0	1	1
F_3	0	0	0	1	0	0	0	1	1	0
F_4	0	0	0	0	1	0	0	1	0	-1

Table 3: A fundamental cycle - edge subset incidence matrix of the directed graph in Figure 1.

Similarly, if we layer the graph using the Longest Path algorithm (see Figure 4(b)), the dummy nodes are assigned to the edge subsets s_0, s_3 and s_8 as follows:

$$\begin{aligned} \mathcal{L}'_+(s_0) &= 1, \\ \mathcal{L}'_+(s_3) &= 2, \text{ and} \\ \mathcal{L}'_-(s_8) &= 1. \end{aligned}$$

It is possible to simultaneously add and remove certain combinations of dummy nodes to any subset of S for which the corresponding multiset of columns in the fundamental cycle - edge subset incidence matrix are linearly dependent over $GF(3)$ so that the fundamental cycles remain balanced and the layering proper. Suppose $\{s'_0, \ldots, s'_k\}$ is such a subset, then there exists some $\lambda_i \in \{-1, 0, 1\}$ $(i = 0, \ldots, k)$ such that

$$\lambda_0 s'_0 + \ldots + \lambda_k s'_k = 0 \text{ (as defined by the columns they label)},$$

and at least one λ_i is non-zero. We can (repeatedly) perform either of the following transformations, assuming there are sufficient dummy nodes on each edge subset and edges in the appropriate directions:

- Either $L'_+(s'_i)+ = \lambda_i$ or $L'_-(s'_i)- = \lambda_i$ $(i = 0, \ldots, k)$.
- Either $L'_+(s'_i)- = \lambda_i$ or $L'_-(s'_i)+ = \lambda_i$ $(i = 0, \ldots, k)$.

The columns of the fundamental cycle - edge subset incidence matrix of the directed graph in Figure 1 (see Table 3) labelled by the edge subsets s_3, s_6, and s_8 are linearly dependent over $GF(3)$ (since $s_3 - s_6 - s_8 = 0$). The coefficients determine whether we can add or remove dummy nodes to the forward or backward edges in the edge subsets. For example, we can transform the Longest Path solution (see Figure 4(b)) into the minimum dummy node (see Figure 4(a)) solution as follows:

- $+s_3$: Remove one 'forward' dummy node from the edge subset s_3 ($\mathcal{L}'_+(s_3)- = 1$).

- $-s_6$: Add one 'forward' dummy node to the edge subset s_6 ($\mathcal{L}'_+(s_6)- = -1$).
- $-s_8$: Remove one 'backward' dummy node from the edge subset s_8 ($L'_-(s_8)+ = -1$).

5 Conclusions

We have described a new ILP formulation for layering a directed acyclic graph with minimum dummy nodes based on a set of fundamental cycles in the underlying undirected graph. A solution describes a family of layerings all with the minimum number of dummy nodes, thus allowing the consideration of other aesthetics by shuffling dummy nodes between edges in a given direction within the same edge subset. We can also transform one solution into another by adding and removing certain combinations of dummy nodes as determined by the coefficients of the dependent subsets of columns in the fundamental cycle - edge subset incidence matrix of the directed graph.

We hope to use this work to find a combinatorial algorithm that can transform an arbitrary layering into one with the minimum number of dummy nodes. As the layerings in Figure 4 show, it is not sufficient to remove all superfluous dummy nodes since we may also need to add dummy nodes to certain edge subsets in order to find the minimum.

References

[1] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing - Algorithms for the Visualization of Graphs.* Prentice Hall, 1999.

[2] E. G. Coffman and R. L. Graham. Optimal scheduling for two processor systems. *Acta Informatica*, 1:200–213, 1972.

[3] E. R. Gansner, E. Koutsofios, S. C. North, and K. Vo. A technique for drawing directed graphs. *IEEE Transactions on Software Engineering*, 19:214–230, 1993.

[4] J. B. Orlin. A faster strongly polynomial minimum cost flow algorithm. *Proceedings of the 20th ACM symposium on theory of computing*, pages 377–387, 1988.

[5] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(2):109–125, 1981.