





A Flow Formulation for Horizontal Coordinate Assignment with Prescribed Width

Michael Jünger¹, Petra Mutzel² , and Christiane Spisla² 

¹ University of Cologne, Cologne, Germany
mjuenger@informatik.uni-koeln.de

² TU Dortmund University, Dortmund, Germany
{petra.mutzel, christiane.spisla}@cs.tu-dortmund.de

Abstract. We consider the coordinate assignment phase of the well known Sugiyama framework for drawing directed graphs in a hierarchical style. The extensive literature in this area has given comparatively little attention to a prescribed width of the drawing. We present a minimum cost flow formulation that supports prescribed width and optionally other criteria like lower and upper bounds on the distance of neighboring nodes in a layer or enforced vertical edge segments. In our experiments we demonstrate that our approach can compete with state-of-the-art algorithms.

Keywords: Hierarchical drawings · Coordinate assignment
Minimum cost flow · Prescribed drawing width

1 Introduction

The Sugiyama framework [12] is a popular approach for drawing directed graphs. It layouts the graph in a hierarchical manner and works in five phases: Cycle removal, layer assignment, crossing minimization, coordinate assignment and edge routing. If the graph is not already acyclic, some edges are reversed to prepare the graph for the next phase. Then each node is assigned to a layer so that all edges point from top to bottom. After that the orderings of the nodes within each layer are determined. In the coordinate assignment phase that we consider here, the exact positions of the nodes are fixed. Finally the edges are layouted, e.g., as straight lines. A good overview over the different phases of the framework can be found in [9].

After the nodes are assigned to layers and the orderings of the nodes within their layers are fixed, the task of the coordinate assignment phase is to compute x -coordinates for all nodes. There are several, sometimes contradicting, objectives in this phase, e.g., short edges, minimum distance between neighboring nodes, straight edges, balanced positions of the nodes between their neighbors in adjacent layers, and few bend points of edges that cross multiple layers. The

criterion “short edges” can be handled by exact algorithms as well as fast heuristics that give pleasant results, possibly also considering other aesthetic criteria.

When it comes to the width of the drawing one usually tries to restrict the maximum number of nodes in one layer, see e.g. [5]. Long edges, i.e. edges that span more than two layers, are often split into paths with one dummy node on each intermediate layer. Healy and Nikolov [8] present a branch-and-cut approach to compute a layering that takes the influence of the number of dummy nodes on the width into account. Jabrayilov et al. [10] do the same in a mixed integer program that treats the first two phases of the Sugiyama framework simultaneously. But still, the maximum number of nodes in one layer does not necessarily define the actual width of the final drawing, as illustrated in Fig. 1. The main objective of most methods for the coordinate assignment phase is “short edges”, which often leads to small drawings, but the width of the final layout is not directly addressed.



Fig. 1. In the left picture the horizontal edge length is $k - 3$ and the width is 1, in the right picture the horizontal edge length is 0 and the width is $k - 2$, where k is the number of layers.

There may be further requirements for the final drawing, such as an aspect ratio in order to make optimal use of the drawing area, or a maximum distance between two nodes on the same layer if they are semantically related. A common request is that inner segments of long edges are drawn as vertical straight lines in order to improve readability.

Related Work. Sugiyama et al. [12] present a quadratic programming formulation that has a combination of two aesthetic criteria as objective function, short edges (closeness to adjacent nodes) and a balanced layout (positioning nodes close to the barycenter of their upper and lower neighbors). Gansner et al. [7] give a simpler formulation in which they replace quadratic terms of the form $(x_v - x_u)^2$ by $|x_v - x_u|$ and leave out the balance terms. The coordinate assignment problem can be interpreted as an instance of the layer assignment problem, and they suggest to apply the network simplex algorithm to an auxiliary graph to obtain a drawing with minimum horizontal edge length. Given an initial layout, some heuristics sweep through the layers and try to shift the nodes to better positions depending on the fixed x -coordinates of their neighbors in adjacent layers, see e.g. [6, 11, 12]. Two fast heuristics that compute coordinates from scratch are

presented by Buchheim et al. [3] and by Brandes and Köpf [2]. Both algorithms draw inner segments of long edges straight and aim for a balanced layout with short edges.

Our Contribution. We formulate the coordinate assignment problem as a minimum cost flow problem that can be solved efficiently. Within this formulation we can fix the maximum width of the final drawing as well as a maximum and minimum horizontal distance between nodes in the same layer and we can enforce straightness to some edges. We compute x -coordinates such that the total horizontal edge length is minimized subject to these further constraints.

2 Notation and Preliminaries

Let $G = (V, E)$ be a directed graph with $|V| = n$ nodes and $|E| = m$ edges. For a directed edge $e = (u, v)$ we denote the start node of e with $start(e) = u$ and the target node of e with $target(e) = v$. A *path* P from u to v of length k is a set of edges $\{e_i = (v_i, v_{i+1}) \mid i = 1, \dots, k \text{ where } u = v_1 \text{ and } v = v_{k+1}\}$. We also write $u \xrightarrow{*} v$. If $v_{k+1} = v_1$ it is called a *cycle*. A graph is called a *directed acyclic graph (DAG)* if it has no cycles. A *layering* \mathcal{L} of a graph assigns every $v \in V$ a *layer* L_i , such that $i < j$ holds for every edge $e = (u, v)$ with $\mathcal{L}(u) = L_i$ and $\mathcal{L}(v) = L_j$. The layering is called *proper* if $\mathcal{L}(v) = \mathcal{L}(u) + 1$ for every edge (u, v) , i.e., the layers of every pair of adjacent nodes are consecutive. An edge that violates the latter property is called a *long edge*. Every graph with a layering can be transformed into a graph with a proper layering by subdividing every long edge into a chain of edges. We denote with $|\mathcal{L}|$ the number of layers and with $|L_i|$ the number of nodes in layer L_i .

An *ordering ord* defines a partial ordering on the nodes of G . For every layer L_i it assigns each node in L_i a number $1 \leq j \leq |L_i|$ and we write $u < v$ if $ord(u) < ord(v)$. We denote with v_j^i the j -th node in layer L_i .

Given a graph G with a layering \mathcal{L} and an ordering *ord* the *horizontal coordinate assignment problem (HCAP)* asks for x -coordinates for every node, so that $x(u) < x(v)$ if $u < v$. We will restrict ourselves to integer coordinates. The *horizontal length* of an edge $e = (u, v)$ is defined as $length(e) = |x(v) - x(u)|$ and the *total horizontal edge length* is $length(E) = \sum_{e \in E} length(e)$. The *width* of the assignment is $\max_{v \in V} x(v) - \min_{v \in V} x(v)$. Unless otherwise stated, we mean the horizontal length whenever we talk about the length of an edge.

$HCAP_{minEL}$ is the variant of HCAP in which we also want to minimize the total horizontal edge length.

We assume familiarity with minimum cost flows. Ahuja et al. [1] give a good overview. Let $N = (V_N, E_N)$ be a directed graph with a super source s and a super sink t , so for all other nodes the amount of incoming flow equals the amount of outgoing flow. We have lower and upper bounds on the edges and a cost function $cost : E_N \rightarrow \mathbb{R}$. Let f be a feasible flow. For a subset of nodes $V' \subseteq V_N \setminus \{s, t\}$ we denote with $f(V') = \sum_{v \in V'} \sum_{e=(v,w)} f(e) = \sum_{v \in V'} \sum_{e=(u,v)} f(e)$ the flow through V' . For s we

define $f(s)$ to be the total amount of flow leaving s . For a subset of edges $E' \subset E_N$ we denote with $f(E') = \sum_{e \in E'} f(e)$ the flow over E' and with $cost(E') = \sum_{e \in E'} cost(e)$ the cost of E' and with $cost_f = \sum_{e \in E_N} f(e) \cdot cost(e)$ the total cost of f .

3 Network Flow Formulation

In this section we describe the construction of a network for the horizontal coordinate assignment problem. Given a minimum cost flow in this network we show how to obtain x -coordinates for all nodes such that the total horizontal edge length is minimized. By a simple modification we can compute x -coordinates that give us minimum total horizontal edge length with respect to a given maximum width of the drawing. The basic idea is that flow represents horizontal distance and we send flow from top to bottom through the layers.

3.1 Network Construction

Let $G = (V, E)$ be a DAG with a proper layering \mathcal{L} and an ordering and let $N = (V_N, E_N)$ be the minimum cost flow network. For now let us assume that neighboring nodes on a layer should have an equal minimum distance of one and that we have no further requirements concerning the edges.

For every layer L_i with $i \in \{1, \dots, |\mathcal{L}|\}$ we add nodes $w_0^i, w_1^i, \dots, w_{|L_i|}^i$ and $z_0^i, z_1^i, \dots, z_{|L_i|}^i$ to N . Imagine the node w_j^i placed above the layer L_i and between v_j^i and v_{j+1}^i (w_0^i is placed at the left end and $w_{|L_i|}^i$ at the right end of the layer). The nodes z_j^i are placed in the same way below layer L_i . Although we do not have a drawing of G at this moment we can still use terms like “above” and “below” because the layering gives us a vertical ordering of the nodes of G and we can talk about “left” and “right” because of the given ordering of the nodes in each layer. Since we are placing the nodes w_j^i and z_j^i “between” the nodes v_j^i and v_{j+1}^i we want to extend the “ $<$ ” relation to give a partial ordering on $V \cup V_N$ in the following way: $w_0^i < v_1^i < w_1^i < v_2^i < \dots < v_{|L_i|}^i < w_{|L_i|}^i$ and $z_0^i < v_1^i < z_1^i < v_2^i < \dots < v_{|L_i|}^i < z_{|L_i|}^i$. We connect w_j^i to z_j^i with an edge a_j^i that has a lower bound of one and an upper bound of ∞ and a cost of zero. The flow over these edges will define the distance between v_j^i and v_{j+1}^i . We denote the set of these edges with A . Figure 2(a) shows an example.

For every layer L_i with $i \in \{1, \dots, |\mathcal{L}|\}$ and every $j \in \{0, \dots, |L_i| - 1\}$ we add edges $\overrightarrow{bw}_j^i = (w_j^i, w_{j+1}^i)$, $\overleftarrow{bw}_j^i = (w_{j+1}^i, w_j^i)$, $\overrightarrow{bz}_j^i = (z_j^i, z_{j+1}^i)$ and $\overleftarrow{bz}_j^i = (z_{j+1}^i, z_j^i)$ to N . The lower bound of these edges is zero and the upper bound is ∞ . The cost of these network edges equals the number of graph edges they “cross over”. That means, the cost of \overleftarrow{bw}_j^i and \overrightarrow{bw}_j^i equals the number of incoming graph edges of node v_j^i and the cost of \overleftarrow{bz}_j^i and \overrightarrow{bz}_j^i equals the number of outgoing graph edges of v_j^i , see Fig. 2(a). Positive flow over one of these edges will cause the crossed-over graph edges to have positive horizontal length. We call the set of these edges B .

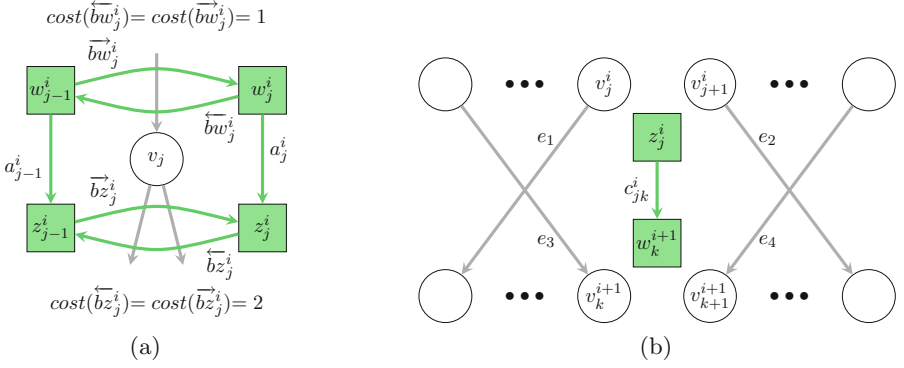


Fig. 2. Illustration of edges of the sets (a) A , B and (b) C . Nodes of G are white circles, nodes of N are green rectangles. Edges of G are gray, edges of N are green. (Color figure online)

Now we connect the nodes of neighboring layers. We could add edges between every z_j^i and every w_k^{i+1} , but we want to keep the number of edges between layers as small as possible. We add edges only in special situations and will show later that this suffices for correctness. For every layer L_i with $i \in \{1, \dots, |\mathcal{L}| - 1\}$ we add edges $c_{00}^i = (z_0^i, w_0^{i+1})$ and $c_{|L_i||L_{i+1}|}^i = (z_{|L_i|}^i, w_{|L_{i+1}|}^{i+1})$ to the network with a lower bound of zero, an upper bound of ∞ and a cost of zero. Additionally we add edges $c_{jk}^i = (z_j^i, w_k^{i+1})$ if there exist $e_1, e_2, e_3, e_4 \in E$ with $start(e_1) = v_j^i$, $start(e_2) = v_{j'}^i$, where $v_{j'}^i$ is the next node to the right of v_j^i with an outgoing edge and $target(e_3) = v_k^{i+1}$, $target(e_4) = v_{k'}^{i+1}$, where $v_{k'}^{i+1}$ is the next node to the right of v_k^{i+1} with an incoming edge and the following conditions holds: $start(e_3) \leq start(e_1) < start(e_2) \leq start(e_4)$ and $target(e_1) \leq target(e_3) < target(e_4) \leq target(e_2)$. We call this situation a *hug* between z_j^i and w_k^{i+1} . These edges get a lower bound of zero, an upper bound of ∞ , and the cost equals the number of graph edges they cross over: $cost(c_{jk}^i) = |\{e = (v_p^i, v_q^i) \in E \mid p \leq j \wedge q \geq k' \vee p \geq j' \wedge q \leq k\}|$. Like the edges of B , flow on edges of this kind will cause horizontal length and we denote the set of all c_{jk}^i by C . Figure 2(b) illustrates a hug situation.

Finally we add a super source s and a super sink t to the network. We connect s with every w_j^1 , $j \in \{1, \dots, |L_1|\}$ and t with every $z_k^{|\mathcal{L}|}$, $k \in \{1, \dots, |L_{|\mathcal{L}|}|\}$. These edges get a lower bound of zero, an upper bound of ∞ and a cost of zero. Figure 3 shows a complete example network. If it is clear from the context which layer or which node is meant, we omit the node subscripts and superscripts.

3.2 Obtaining Coordinates and Correctness

Let f be a feasible flow in the network described above. We observe that $f(a_j^i) = f(w_j^i) = f(z_j^i)$ since a_j^i is the only outgoing edge of w_j^i and the only incoming

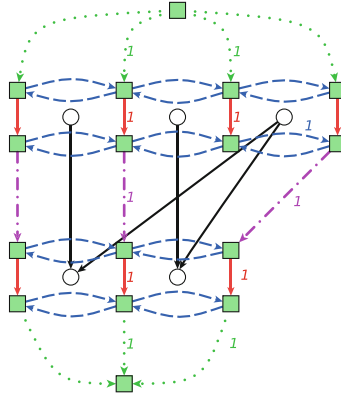


Fig. 3. An example network with underlying graph. Edges of the set A are red (solid), edges of the set B are blue (dashed) and edges of the set C are purple (dashdotted). The numbers along the edges denote the flow, unlabeled edges carry no flow. (Color figure online)

edge of z_j^i . We define the x -coordinate of a node v_j^i as

$$x(v_j^i) := \sum_{l=0}^{j-1} f(a_l^i) = \sum_{l=0}^{j-1} f(w_l^i) = \sum_{l=0}^{j-1} f(z_l^i). \tag{1}$$

Together with $y(v_j^i) = i$ we get an induced drawing with a feasible coordinate assignment, because for every v_j, v_k within the same layer $x(v_j) < x(v_k)$ if and only if $v_j < v_k$ (since the amount of flow over edges $a \in A$ is always positive).

Now we want to explain the correspondence between the cost of a flow f and the total horizontal edge length of the resulting drawing. The intuition is, that if flow is sent from the right of $start(e)$ to the left of $target(e)$ for some edge e , then $target(e)$ is “pushed” to the right because of the additional flow on the left. This results in a horizontal expansion of e . We define for an edge $e = (u, v) \in E$

$$\begin{aligned} \vec{E}(e) := & \{bw \in B \mid start(bw) < v \wedge target(bw) > v\} \\ & \cup \{bz \in B \mid start(bz) < u \wedge target(bz) > u\} \\ & \cup \{c \in C \mid start(c) < u \wedge target(c) > v\} \end{aligned}$$

as the set of network edges that start to the left of e and end to the right of e , thus cross over e from left to right. Analogously the set of network edges that cross over a graph edge from right to left is

$$\begin{aligned} \overleftarrow{E}(e) := & \{bw \in B \mid start(bw) > v \wedge target(bw) < v\} \\ & \cup \{bz \in B \mid start(bz) > u \wedge target(bz) < u\} \\ & \cup \{c \in C \mid start(c) > u \wedge target(c) < v\}. \end{aligned}$$

We make the following observations:

Property 1. $cost(g) = |\{e \in E \mid g \in \vec{E}(e)\}| + |\{e \in E \mid g \in \overleftarrow{E}(e)\}| \forall g \in B \cup C$.

Property 2. *Because of the flow conservation rule we have*

$$\begin{aligned} \sum_{j=0}^{|L_i|} f(w_j^i) &= \sum_{j=0}^{|L_i|} f(z_j^i) \text{ for all } i \in \{1, \dots, |\mathcal{L}|\} \text{ and} \\ \sum_{j=0}^{|L_i|} f(w_j^i) &= \sum_{j=0}^{|L_k|} f(w_j^k) = f(s) \text{ for all } i, k \in \{1, \dots, |\mathcal{L}|\}. \end{aligned}$$

Property 3. *The width of the induced drawing is*

$$\max_{1 \leq i \leq |\mathcal{L}|} \left(\sum_{j=1}^{|L_i|-1} f(w_j^i) \right) \leq f(s).$$

Property 4. *Let $e = (v_j^i, v_k^{i+1})$ be an edge. Then*

$$\sum_{w_i^{i+1} < v_k^{i+1}} f(w_i^{i+1}) = \sum_{z_l^i < v_j^i} f(z_l^i) + f(\overleftarrow{E}(e)) - f(\vec{E}(e)).$$

The last property is illustrated in Fig. 4. The total flow that reaches all w_j^{i+1} that are to left of $target(e)$ comes from the z_j^i that are to the left of $start(e)$ and from nodes that are to the right of $target(e)$ or $start(e)$. Flow from the latter nodes has to pass over e from right to left. Flow from a node z_j^i that is to the left of $start(e)$ and does not enter one of the w_j^{i+1} left of $target(e)$ has to pass over e from left to right.

Lemma 1. *For a feasible flow f and the induced drawing $cost_f \geq length(E)$ holds.*

Proof. Let $e = (v_j^i, v_k^{i+1})$ be an edge of G . The length of e is $length(e) = |x(v_j^i) - x(v_k^{i+1})|$ and together with (1) we have

$$\begin{aligned} length(e) &= \left| \sum_{l=0}^{j-1} f(z_l^i) - \sum_{l=0}^{k-1} f(w_l^{i+1}) \right| \\ &= \left| \sum_{z_l^i < start(e)} f(z_l^i) - \sum_{w_l^{i+1} < target(e)} f(w_l^{i+1}) \right| \\ &= \left| f(\vec{E}(e)) - f(\overleftarrow{E}(e)) \right| \quad (\text{by Property 4}). \end{aligned}$$

Therefore we have for the total edge length

$$\begin{aligned} length(E) &= \sum_{e \in E} \left| f(\vec{E}(e)) - f(\overleftarrow{E}(e)) \right| \\ &\leq \sum_{e \in E} \left(\left| f(\vec{E}(e)) \right| + \left| f(\overleftarrow{E}(e)) \right| \right) \\ &= \sum_{e \in E} \left(f(\vec{E}(e)) + f(\overleftarrow{E}(e)) \right) \\ &= \sum_{g \in E_N} f(g) \cdot |\{e \in E \mid g \in \vec{E}(e)\} \cup \{e \in E \mid g \in \overleftarrow{E}(e)\}| \\ &= \sum_{g \in E_N} f(g) \cdot cost(g) \quad (\text{by Property 1}) \\ &= cost_f. \end{aligned}$$

□

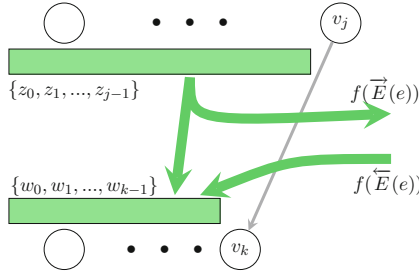


Fig. 4. Illustration of Property 4. The rectangles represent all network nodes to the left of v_j and v_k , respectively. The thick arrows represent the flow of several edges.

Lemma 2. *Let Γ be a drawing of G . There exists a flow f that induces Γ and whose cost is equal to the total edge length of Γ .*

Proof. If necessary, we set $x(v) := x(v) - \min_{v \in V} x(v)$ so that the smallest x -coordinate is zero. That gives us an equivalent drawing. We construct the flow f as follows: Let ω be the width of Γ . We send ω units of flow from s to t , so that the k -th unit takes the path $P_k = s \xrightarrow{*} w_{j_1}^1 \xrightarrow{*} w_{j_2}^2 \xrightarrow{*} \dots \xrightarrow{*} w_{j_{|\mathcal{L}|}}^{|\mathcal{L}|} \xrightarrow{*} t$, where $w_{j_i}^i$ is chosen so that $x(v_{j_{i+1}}^i) \geq k$ and $x(v_{j_i}^i) < k$ ($w_{j_i}^i = w_0^i$, if $x(v_1^i) \geq k$ and $w_{j_i}^i = w_{|L_i|}^i$, if $x(v_{|L_i|}^i) < k$). That means we send the k -th unit through the k -th “column” of Γ . This is always possible, because of the subpaths $w_{j_i}^i \rightarrow z_{j_i}^i \xrightarrow{*} z_0^i \rightarrow w_0^{i+1} \xrightarrow{*} w_{j_{i+1}}^{i+1}$. So for every v there are $x(v)$ units of flow that pass by to the left of v , thus giving us correct coordinates for all nodes.

We define $E_k^i := \{e = (v_j^i, v_l^{i+1}) \in E \mid x(v_j^i) < k \text{ and } x(v_l^{i+1}) \geq k\} \cup \{e \in E \mid x(v_j^i) \geq k \text{ and } x(v_l^{i+1}) < k\}$, i.e. all edges that cross over the k -th column between L_i and L_{i+1} . We show that there exists a path P_k that produces the same cost as the number of graph edges that cross over the k -th column in total, that is $cost(P_k) = \sum_{i=1}^{|\mathcal{L}|-1} |E_k^i|$. Then we have $\sum_{k=1}^{\omega} cost(P_k) = \sum_{k=1}^{\omega} \sum_{i=1}^{|\mathcal{L}|-1} |E_k^i| = length(E)$ and we have proven the lemma.

It suffices to focus on the subpath P_k^i from $z = z_{j_i}^i$ to $w = w_{j_{i+1}}^{i+1}$ between two consecutive layers. Notice that network edges (s, w^1) , (w^i, z^i) and $(z^{|\mathcal{L}|}, t)$ do not contribute to the cost of the flow. For better readability we denote the nodes of L_i with u_j and the nodes of L_{i+1} with v_j and we omit the superscripts. If not stated otherwise we use z_j for z_j^i and w_j for w_j^{i+1} . We construct $P' = P_k^i$ so that $cost(P') = |E'| = |E_k^i|$.

Case 1: There exists no edge e with $start(e) < z$ and $target(e) < w$.

That means every edge e with $start(e) < z$ has $target(e) > w$, and if $target(e) < w$ then $start(e) > z$. Then we set $P' = z \rightarrow z_{j_{i-1}} \xrightarrow{*} z_0 \rightarrow w_0 \rightarrow w_1 \xrightarrow{*} w$. For every $u_j < z$ with p outgoing edges P' uses exactly one \overleftarrow{bz} with cost p . All these edges are in E' . For every $v_j < w$ with q incoming edges we use exactly one \overrightarrow{bw}

with cost q . Again these edges are in E' . So $cost(P') = |E'|$, since there are no other edges in E' .

Case 2: There exists no edge e with $start(e) > z$ and $target(e) > w$.

Arguing like in Case 1, we set $P' = z \xrightarrow{*} z_{|L_i|} \rightarrow w_{|L_{i+1}|} \xrightarrow{*} w$. As before the cost of P' equals $|E'|$.

Case 3: There exists an edge e_l with $start(e_l) < z$ and $target(e_l) < w$ and another edge e_r with $start(e_r) > z$ and $target(e_r) > w$.

Let e_l be the edge with the biggest $x(start(e))$ of all edges e with $start(e) < z$ and $target(e) < w$, and let e_r be the edge with the smallest $x(start(e))$ of all edges e with $start(e) > z$ and $target(e) > w$.

Case 3.1: There is at least one node u' with outgoing edges and $start(e_l) < u' < z$.

Let $u_g = start(e_l)$ and $v_{g'} = target(e_l)$. We know $v_{g'} < w$. Let $v_{h'}$ be the first node to the right of $v_{g'}$ with an edge $e_{r'} = (u_h, v_{h'})$ and $u_h > u_g$. Such a node does exist, since we have e_r . Notice that $v_{h'}$ might be to the right of w .

Then we have a hug: Set $e_1 = e_l$, set e_2 to one outgoing edge of u_{g+1} (or the next node to the right of u_g , which has an outgoing edge), $e_4 = e_{r'}$ and set e_3 to one incoming edge of $v_{h'-1}$ (or the next one to the left of $v_{h'}$), see Fig. 5. Notice that e_1 may coincide with e_3 and e_2 with e_4 .

We have $start(e_3) \leq start(e_1)$, because we chose $e_1 = e_l$ with the biggest $x(start(e))$ and $v_{h'}$ is the first node to the right of $v_{g'}$ with an adjacent node to the right of u_g . So every node between $v_{g'} = target(e_1)$ and $v_{h'}$, including $v_{h'-1} = target(e_3)$, can only have adjacent nodes to the left of $u_g = start(e_1)$. It is clear that $start(e_1) < start(e_2)$ and $start(e_2) \leq start(e_4)$, since $start(e_4) = u_h > u_g$. By choice of e_1 , e_3 and e_4 $target(e_1) \leq target(e_3) < target(e_4)$ holds. We know that $target(e_2) > w$ because there is at least one node between $start(e_1)$ and z whose outgoing edges have to end to the right of w because of the choice of e_1 . If $target(e_4) > target(e_2)$ then e_2 would have been chosen for $e_{r'}$ and therefore for e_4 . So $target(e_4) \leq target(e_2)$ also holds. So there exists $c_{g(h'-1)} \in E_N$ and we set $P' = z \xrightarrow{*} z_g \rightarrow w_{h'-1} \xrightarrow{*} w$.

Now for the cost. A subset of E' are the edges e with $z_g < start(e) < z$ and $target(e) > w$, which are covered by the \overleftarrow{bz} of P' .

We have two options. First, if $w_{h-1} > w$ then all edges e with $start(e) < z_g$ and $target(e) > w_{h'-1}$ are covered by $c_{g(h'-1)}$ and the remaining edges e with $start(e) < z_g$ and $w < target(e) < w_{h'-1}$ are covered by the \overleftarrow{bw} of P' . Edges e with $start(e) > z > u_g$ and $target(e) < w < w_{h'}$ are also covered by $c_{g(h'-1)}$. There cannot be any edge e with $z < start(e)$ and $w < target(e) < w_{h'-1}$ or $z_g < start(e) < z$ and $target(e) < w$, which would be crossed over by two different edges of P' , due to the choice of edges e_1 to e_4 .

Second, if $w_{h-1} < w$ then $c_{g(h'-1)}$ covers all edges e with $start(e) < z_g$ and $target(e) > w > w_{h'-1}$ and all edges e with $start(e) > z > z_g$ and $target(e) < w_{h'-1} < w$. Edges e with $start(e) > z$ and $w_{h'-1} < target(e) < w$ are covered by the \overrightarrow{bw} . Again there are no edges that are crossed over twice by P' due to the

choice of e_1 to e_4 . And there are no edges in E' that are not covered by some edge of P' .

Case 3.2: $start(e_l)$ is the next node to the left of z with outgoing edges, but there is at least one node u' with outgoing edges and $start(e_r) > u' > z$.

This case is analogous to Case 3.1.

Case 3.3: $start(e_l)$ is the next node to the left of z with outgoing edges and $start(e_r)$ is the next node to the right of z with outgoing edges.

Let $e_l = (u_g, v_{g'})$ and $v_{h'}$ be the first node right of $v_{g'}$ with an adjacent node $u_h > u_g$. Again we have a hug. Set $e_1 = e_l$, $e_2 = e_r$, e_3 to an incoming edge of $v_{h'-1}$ (or a lower node, if necessary) and $e_4 = (u_h, v_{h'})$.

With the same arguments as in Case 3.1 we convince ourselves that e_1 , e_2 , e_3 and e_4 are indeed a hug and we have $c_{j_i(h'-1)}$. We set $P' = z \rightarrow w_{h'-1} \xrightarrow{*} w$. As before $cost(P') = |E'|$. □

Theorem 1. *A minimum cost flow in the network described above solves $HCAP_{minEL}$.*

Proof. Lemma 1 and Lemma 2. □

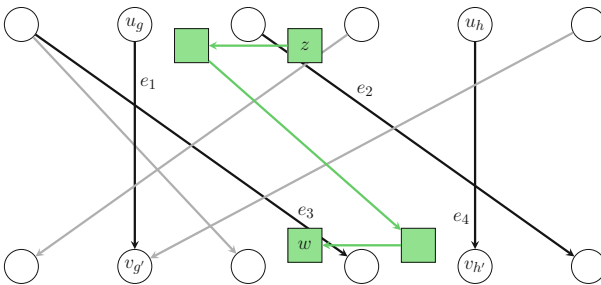


Fig. 5. Case 3.1. Only relevant network nodes and edges are depicted. Edges that participate in the hug are black.

For controlling the maximum width of the drawing we make use of Property 3, which states that the width of the drawing is at most the flow leaving s . We can add an additional node s' and an edge (s, s') to N and replace all edges of the form (s, w_j^1) with (s', w_j^1) . Now we can limit the maximum width of the drawing by setting the upper bound of (s, s') to an appropriate value.

Further constraints can be modelled by manipulating the network. By adjusting the lower and upper bounds of edges $a \in A$ we can realize minimum and maximum distances between two neighboring nodes on the same layer. By removing every $g \in \overleftarrow{E}(e) \cup \overrightarrow{E}(e)$ from the network, we can enforce the edge e to be drawn vertically.

4 Experimental Results

In our experiment we want to demonstrate that we are able to restrict the width of the drawing without paying too much in terms of total (horizontal) edge length and time.

We implemented the algorithm from Sect. 3, which we will call MCF within the Open Graph Drawing Framework [4] (OGDF) and used the OGDF network simplex software to solve the minimum cost flow problem. We also implemented the approach of Gansner et al. [7] (Gansner) that also uses the network simplex algorithm. Additionally we use three other OGDF methods: an ILP that also takes balancing the nodes between their neighbors into account (LP), the algorithm of Buchheim, Jünger and Leipert [3] (BJL) and the algorithm of Brandes and Köpf [2] (BK). All algorithms draw inner segments of long edges as vertical lines, since this is generally desirable for good readability. MCF is configured to compute a layout with minimum edge length with respect to minimum possible width and Gansner computes coordinates that minimize the total edge length regardless of width. We used a subset of the AT&T graphs from www.graphdrawing.org/data.html consisting of 1277 graphs with 10 to 100 nodes as our test set.

The test was run on an Intel Xeon E5-2640v3 2.6 GHz CPU with 128 GB RAM.

Figures 6, 7, and 8 show the results. The whiskers in Figs. 6 and 7 cover 95% of the data and outliers are omitted for better readability. Figure 8 shows absolute values for MCF and Fig. 9 displays three example drawings.

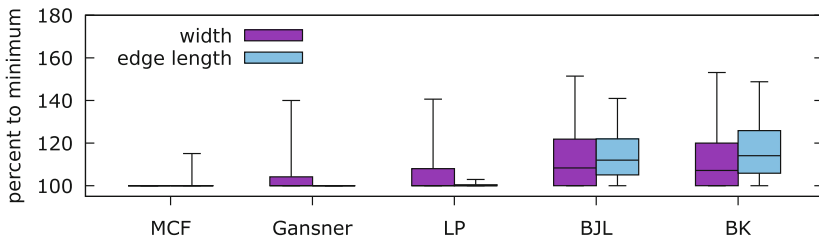


Fig. 6. Width and total edge length produced by MCF, Gansner, LP, BJJ and BK relative to minimum width, resp. edge length.

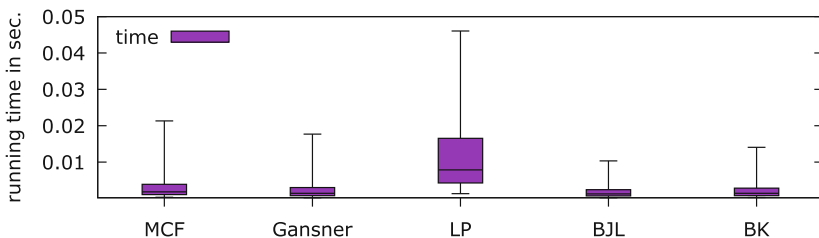


Fig. 7. Running time for MCF, Gansner, LP, BJJ and BK.

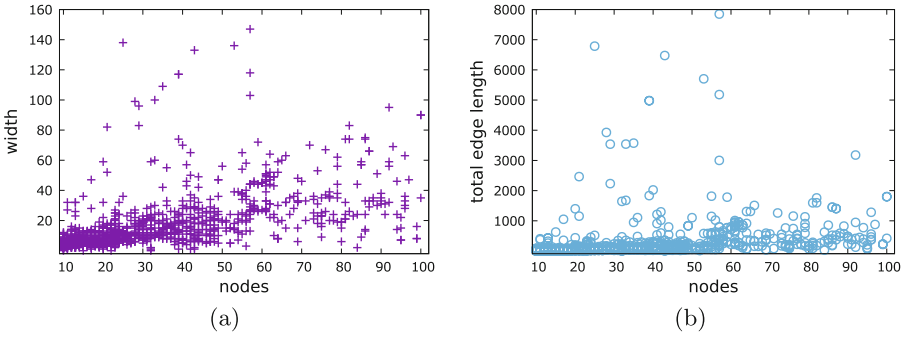


Fig. 8. Absolute values of (a) width and (b) total edge length for MCF.

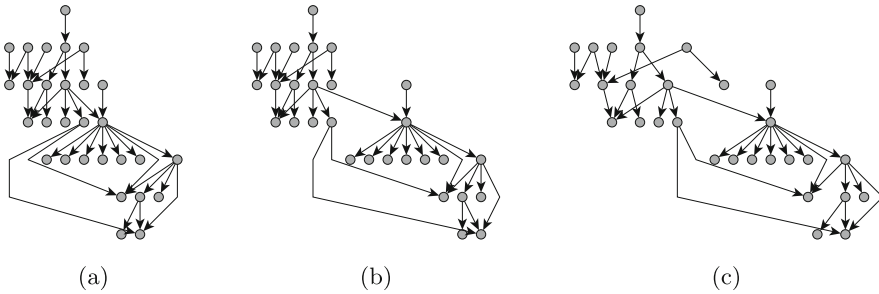


Fig. 9. Example drawings of a graph with 29 nodes and 33 edges. (a) MCF: width: 9, edge length: 58. (b) Gansner: width: 13, edge length: 54. (c) BK: width: 16.5, edge length: 63.5.

In Fig. 6 the resulting total edge length and width of the drawings are depicted relative to the minima that are computed by Gansner and MCF, respectively. We see that MCF still achieves good results in terms of total edge length, even though it has the restriction of meeting the minimum width. The total edge length of drawings computed with MCF is on average 2.2% over the minimum, while drawings produced with Gansner have on average a width that is 8.9% over the minimum. In an extreme example with minimum width 1, Gansner results in width 15.

Figure 7 shows the running time in seconds. MCF (4.9ms on average) is a bit slower than Gansner (3.9ms on average). The fastest algorithm on average is BJL with 2.5 ms.

5 Conclusion

We presented a minimum cost flow formulation for the coordinate assignment problem that minimizes the total edge length with respect to several optional criteria like the maximum width or lower and upper bounds on the distance of

neighboring nodes in a layer. In our experiments we showed that our approach can compete with state-of-the-art algorithms.

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: *Network Flows - Theory, Algorithms and Applications*. Prentice Hall, Upper Saddle River (1993)
2. Brandes, U., Köpf, B.: Fast and simple horizontal coordinate assignment. In: Mutzel, P., Jünger, M., Leipert, S. (eds.) *GD 2001*. LNCS, vol. 2265, pp. 31–44. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45848-4_3
3. Buchheim, C., Jünger, M., Leipert, S.: A fast layout algorithm for k -level graphs. In: Marks, J. (ed.) *GD 2000*. LNCS, vol. 1984, pp. 229–240. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44541-2_22
4. Chimani, M., Gutwenger, C., Jünger, M., Klau, G.W., Klein, K., Mutzel, P.: The open graph drawing framework (OGDF). In: Tamassia, R. (ed.) *Handbook on Graph Drawing and Visualization*, pp. 543–569. Chapman and Hall/CRC, Boca Raton (2013)
5. Coffman, E.G., Graham, R.L.: Optimal scheduling for two-processor systems. *Acta Informatica* **1**(3), 200–213 (1972). <https://doi.org/10.1007/BF00288685>
6. Eades, P., Lin, X., Tamassia, R.: An algorithm for drawing a hierarchical graph. *Int. J. Comput. Geom. Appl.* **6**(2), 145–156 (1996). <https://doi.org/10.1142/S0218195996000101>
7. Gansner, E.R., Koutsofios, E., North, S.C., Vo, K.P.: A technique for drawing directed graphs. *Softw. Eng.* **19**(3), 214–230 (1993). <https://doi.org/10.1109/32.221135>
8. Healy, P., Nikolov, N.S.: A branch-and-cut approach to the directed acyclic graph layering problem. In: Goodrich, M.T., Kobourov, S.G. (eds.) *GD 2002*. LNCS, vol. 2528, pp. 98–109. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-36151-0_10
9. Healy, P., Nikolov, N.S.: Hierarchical drawing algorithms. In: Tamassia, R. (ed.) *Handbook on Graph Drawing and Visualization*, pp. 409–453. Chapman and Hall/CRC, Boca Raton (2013)
10. Jabrayilov, A., Mallach, S., Mutzel, P., Rüegg, U., von Hanxleden, R.: Compact layered drawings of general directed graphs. In: Hu, Y., Nöllenburg, M. (eds.) *GD 2016*. LNCS, vol. 9801, pp. 209–221. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50106-2_17
11. Sander, G.: A fast heuristic for hierarchical Manhattan layout. In: Brandenburg, F.J. (ed.) *GD 1995*. LNCS, vol. 1027, pp. 447–458. Springer, Heidelberg (1996). <https://doi.org/10.1007/BFb0021828>
12. Sugiyama, K., Tagawa, S., Toda, M.: Methods for visual understanding of hierarchical system structures. *IEEE Trans. Syst. Man Cybern.* **11**(2), 109–125 (1981). <https://doi.org/10.1109/TSMC.1981.4308636>