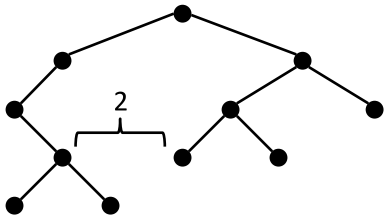
1.1 Layered Tree Algorithm



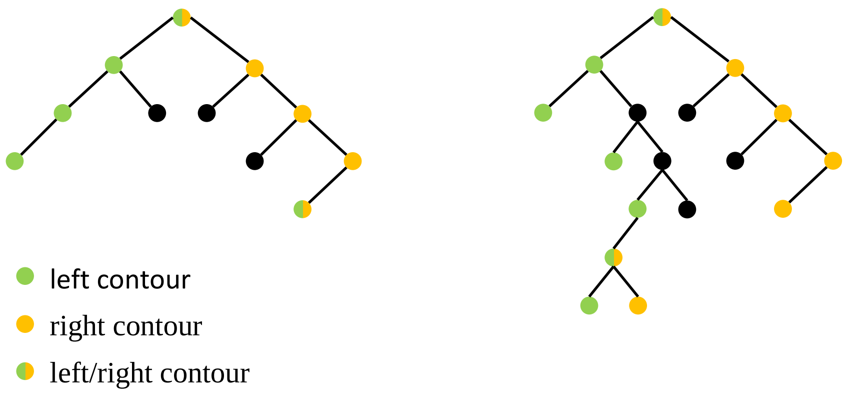
THEOREM
Let T be a Binary Tree with n vertices. Algorithem 1 constructs a drawing $T'$of T in $O(n)$ time such that T' is layered, in example the y-coordinate of each vertex is equal to minus depth of the vertex.

- T' is planar, straightline and strictly drawn
- T' is embedding the presaving in example the left-to-right order of each vertex is preserved
- Any two vertices of T' have horizontal and vertical distance of at least 1
- The x-coordinates of a parent with 2 children is the average of the x-coordinate of its children
- The area of T' is $O(n^2)$

IMPLEMENTATION OF ALGORITHM 1

1. Postorder traveling of T to recursively compute for each vertex v the horizon displacement of its children with respect to v
2. Preorder traveling of T to compute x-coordinates (accumulation of displacements on path to root) and y-coordinates (depth) of all vertices

DEFINITION
The left contour of a Binary Tree T with height h is the sequenze of vertices $v_0, \dots, v_n$ such that v_i is the leftmost vertex of T with depth i. Right contour is symmetrically defined.

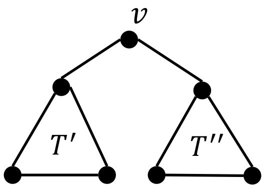


3. Follow left contour of right subtree and right contour of left subtree. ∀ vertices in postorder traveling invariant. After completing processing vertiex v, the right contour and left contour of the subtrees rooted at v are stored in linked lists

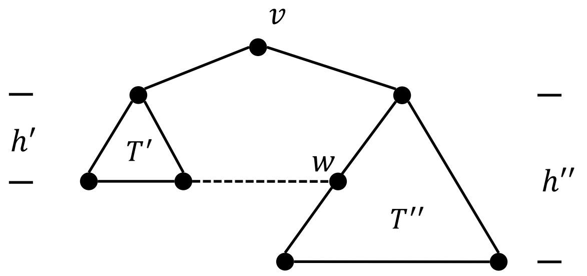
Constructing the contour lists

Let $T(u)$ be the subtree rooted at v and T', T'' be the left and right subtrees of v.

1. If T' and T'' have the same height, then the left contour list of $T(v)$ is the left contour of T' plus v; the right conture of $T(v)$ is the same as the right contour of T'' plus v.



2. If the height of $T' < T''$ then the right conture of $T(v)$ ist the same as the right contour of T'' plus v. Let h' be the height of T'; u the bottemmost vertex on the left contour of T'. Let w be the vertex of the left contour of T'' with depht $h' + 1$. Then the left contour of $T(v)$ consists of v, the left contour of T' and the partion oft he left contour of T'' beginning at w and ending at total height h'' of T''.



3. Height of $T' > T''$ symmetric to 2.

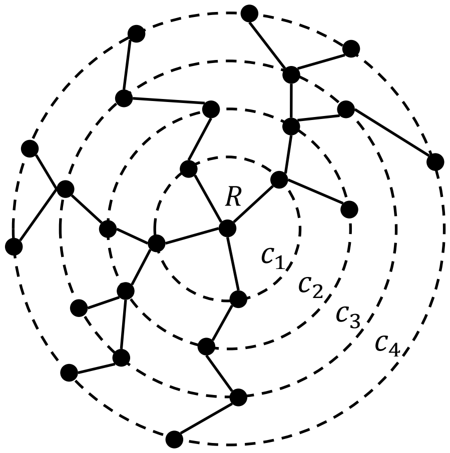
⇒ Must travel down the contours of T', T'' only as far as the height of the subtree with smaller height.

⇒ The time spent processing vertex v in the postorder travel is proportional to the minimum of the heights T', T''.

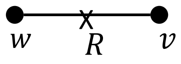
Running time of postorder travel.:

$$\sum_{c \in T} (1 + \min\{h'(v), h''(v)\}) =$$
$$n + \sum_{v \in T} \min\{h'(v), h''(v)\} = O(n)$$

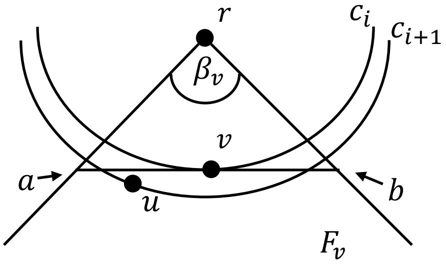
1.2 Radial drawings



Find the root: pruning
1 vertex left → root
2 vertices left:



Definition
A radial drawing is a variation of a layered drawing where the root of the tree is placed at the origin and layers are concentric circles centered at the origin.



The children of v are arranged on $c_i + 1$ according to the number of leaves in their resprective subtrees. That means for each child of a of v the angle β_u of W_u is $\beta_u = \min(\frac{l(u)*\beta_v}{l(v)}, \tau)$

1.3 Planarity testing

Observation

- A graph is planar if all its connected components are planar
- A connected graph is planar, iff all its biconnected components are planar

⇒

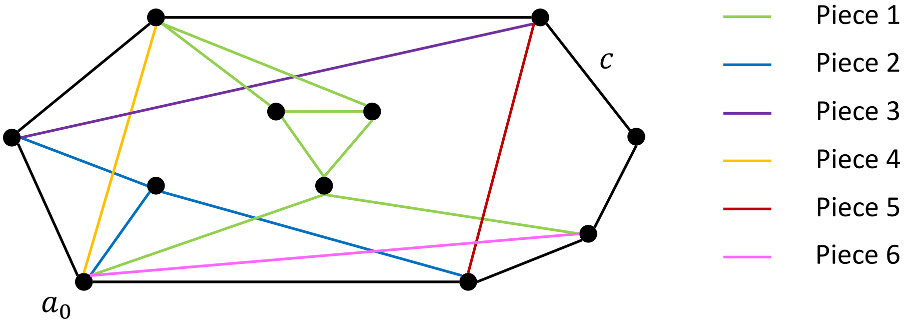
1. Eular planar?
2. Decompose graph in connected and biconnected components

⇒ Algorithm works on biconnected components.

Definition

Let G be a biconnected graph given a cycle C of G, we partition the edges of $G \backslash C$ into classes:

Two edges are in the same class if there is a path between them that does not contain any vertex of C. The subgraph induced by edges in one class is a piece of G w.r.t. C.



Two types of pieces:

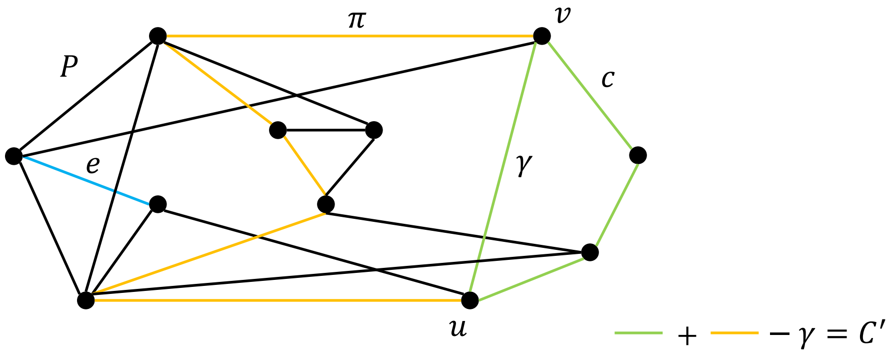
- Single edge between two vertices of C (P_3, P_4, P_5, P_6)
- Connected graph with at least 1 vertex not in C (P_1, P_2)

Definition

- The vertices of a piece P that are in C are called attachments of P
- A cycle C of G is separating, if it has at least two pieces and nonseperating if it has none.
- If $G = C$ then C has no pieces

Lemma

Let G be a biconnected graph and let C be a nonseperating cycle of G with piece P. If P is not a path, then G has a seperaiting cycle C' consisting of a subgraph of C plus a path of P between two attachments.

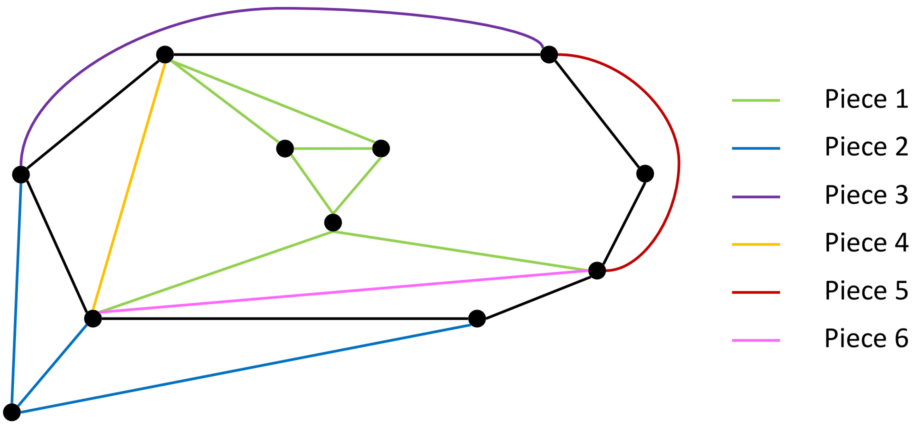


Proof

Let u, v be two attachments of P that are consecutive in the circular ordering and let γ be a subpath of C between u and v that does not contain any attachments of C. P is connected

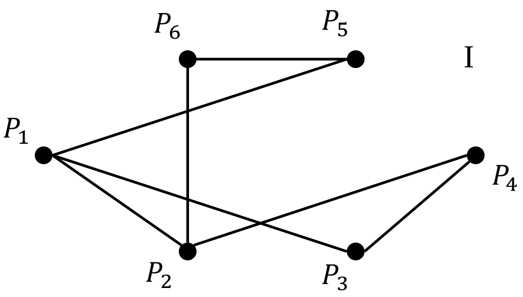
⇒ ∃ path π in P between u and v. Be the cycle obtained by replacing γ with π. If P is not a path let c be an edge of P not π. There is a piece of C' containing c that is constructed from γ. ⇒ C' has two pieces. ■

If G is planar, then in any planar drawing of G each piece is drawn either completely inside or outside of C.



Definition
Two pieces of G w.r.t. C interlace if they cannot be drawn on the same side of C without intersection.

Definition
The interlacement graph I of the pieces of G w.r.t. C is the graph with vertices representing the pieces and the edges of interlacing pieces.



Observation
If G is planar, then I of the pieces w.r.t. cycle C must be bipartite.

⇒ Interlacing pieces must be drawn on opposite sides of C.

Theorem

A biconnected graph G with a cycle C is planar iff it holds that:

- For each piece P of G w.r.t. C the graph obtained by adding P to C is planar
- The interlacement graph I of the pieces of G w.r.t. C is bipartite

Algorithm 2 – Planarity testing

Input:

- Biconnected graph with n vertices and edges e: $e \leq 3n - 6$
- Separating cycle c

Output:

- Graph planar? yes/no

Algorithm:

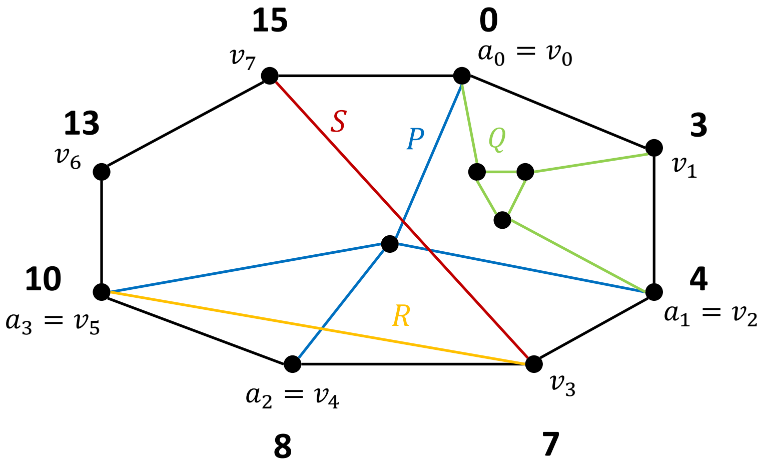
```
1. Compute pieces of $G$ w.r.t. $C$
2. For each piece $P$ that is not a path:
      a. Let $P'$ be the graph obtained by adding $P$ to $C$
      b. Let $C'$ be the cycle of $P'$ obtained from $C$ by replacing the portion of $C$ between two consecutive attachements with a path of $P$ between them
      c. Apply Algorithm 2 recursively to $P'$ and cycle $C'$
3. Compute interlacement graph $I$ of the pieces
4. Test whether $I$ is bipartite. If not return "not planar"
5. Return "planar"
```

According to step 3 – Interlacement Graph

Given a piece P with attachements $a_0, \dots, a_{k-1}$ in this order around C, we label the vertices of C with integers in the range of $[0, 2n - 1]$:

- Label $2 * i$ if it is an attachement of P
- Label $2 * i + 1$ otherwise

A piece Q does not interlace with P iff all its attachements have labels that lie in the open interval between two consecutive even numbers.



- Labeling of C: $O(n)$ time
- Testing whether Q interlaces with P takes time proportional to the number of attachements of $Q \leq 1 + |E(Q)|$; all pieces are edge-disjoint. → linear time

⇒ Step 3 takes $O(n^2)$ time

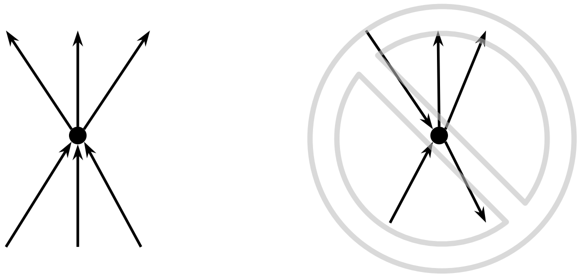
Step 4 has $O(n)$ and $O(n^2)$ edges. Test for bipartiteness takes $O(n^2)$ time.

⇒ 1 recursive call: $O(n^2)$ time
⇒ $O(n)$ calls
⇒ Total running time of Algorithm 2: $O(n^3)$

2.1 Planar orientations

Lemma
Let G be a planar st-graph. Each face of G consists of two directed paths with common origin $[\text{orig}(f)]$ and destination $[\text{dest}(f)]$. The following property is "dual" to the one above.

Lemma
For each vertex v of a planar st-graph, the incoming edges appear consecutively around v, and so do the outgoing edges.



Lemma

For any two faces f, g of a planar st-graph exactly one of the following holds:

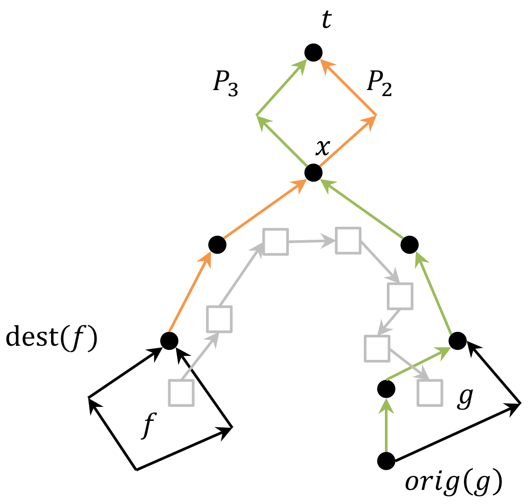
- G has a directed path from $dest(f) \rightarrow orign(g)$
- G has a directed path from $dest(g) \rightarrow orig(f)$
- G^* has a directed path from $f \rightarrow g$
- G^* has a directed path from $g \rightarrow f$

Proof

Consider topological sorting of G; w.l.o.g. (without loss of generality) assume $\#\mathrm{dest}(f) < \#\mathrm{orign}(g)$. We call a path from a vertex $v \in G$ that always takes the leftmost/rightmost outgoing edge the leftmost/rightmost path from v

- P_1 leftmost path from $\mathrm{dest}(f) \rightarrow t$
- P_2 rightmost path from $\mathrm{dest}(f) \rightarrow t$
- P_3 leftmost path from $\mathrm{dest}(g) \rightarrow t$
- P_4 rightmost path from $\mathrm{dest}(g) \rightarrow t$

if there is a directed path from $\mathrm{dest}(f) \rightarrow \mathrm{orig}(g)$. Otherwise, either P_2 crosses P_3 or P_1 crosses P_4 at a common vertex x.



x: First vertex at witch P_2, P_3 intersect

□→□ Directed path from f to g

Then every edge incident to any vertex in P_2 from the right side is incoming and every edge incident to any vertex in P_3 from the left side is incoming.

⇒ Because of the construction of G^* there is a directed path in G^* $f \rightarrow g$ ■

An element of $V \cup E \cup F$ in G is an object of G. For a vertex v:

- $\text{orig}(v) = \text{dest}(v) = v$

For a face f:

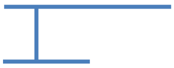
- $\text{left}(f) = \text{right}(f) = f$

LEMMA
For any two objects o_1, o_2, o_3 of a planar st-graph, exactly one of the following holds:

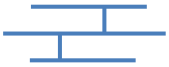
- G has a directed path from $\text{dest}(o_1) \rightarrow \text{orig}(o_2)$
- G has a directed path from $\text{dest}(o_2) \rightarrow \text{orig}(o_1)$
- G^* has a directed path from $\text{right}(o_1) \rightarrow \text{left}(o_2)$
- G^* has a directed path from $\text{right}(o_2) \rightarrow \text{left}(o_1)$

2.2 Constrained visibility representations

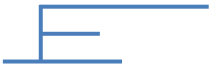
DEFINITION
Two horizontal segments are visible if they can be joined by a vertical segment that does not intersect any other horizontal segment.



visible



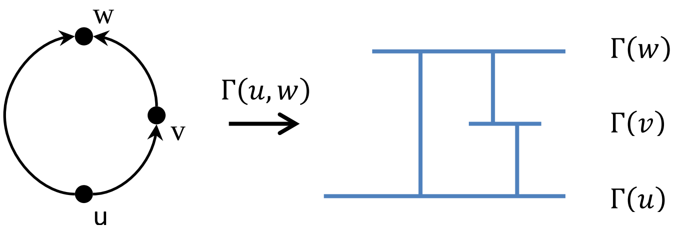
not visible



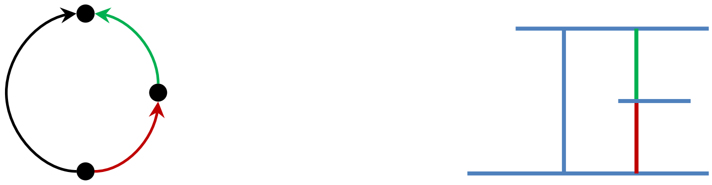
visible

DEFINITION
Let G be a planar st-graph, H a visibility representation Γ of G draws each vertex v as horizontal vertex segment $\Gamma(v)$ and each edge (u, v) as vertical edge segment $\Gamma(u, v)$ such that:

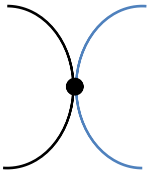
- Vertex segments do not overlap
- Edge segments do not overlap
- An edge segment $\Gamma(u, v)$ has its bottom endpoint on $\Gamma(u)$. Its top endpoint on $\Gamma(v)$ and does not intersect any other vertex segments.



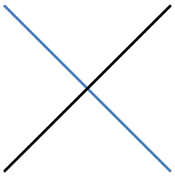
DEFINITION
A visible representation is constrained if some predefined edges have the same x-coordinates.



DEFINITION
Let G be a planar st-graph with n vertices. Two parts π_1, π_2 of G are nonintersecting if they are edge disjoint and do not cross at a common vertex.



noncrossing



crossing

ALGORITHM 3 – CONSTRAINED VISIBILITY
Input:

- Planar st-graph G with vertices
- A set Π of nonintersecting Gpaths covering $E(G)$

Output:

- Constrained visibility representation Γ of G with integer coordinates and area $O(n^2)$

1. Construct G_Π with
    - Vertex set $F \cup \Pi$
    - Edge set $\{(f,\pi) | f = \text{left}(e) \text{for some edge}$

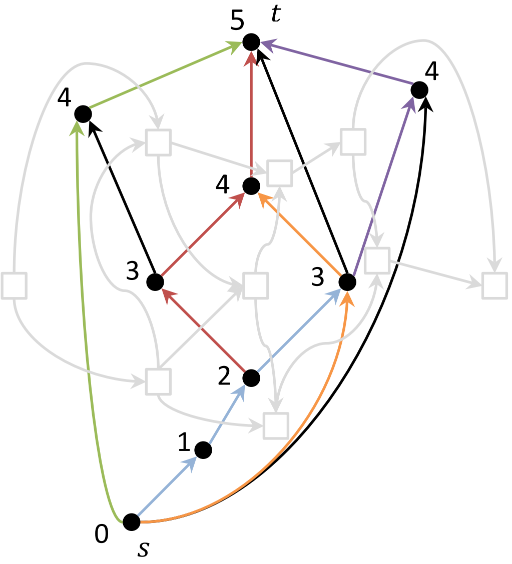
e of path $\pi\} \cup \{(f,\pi) | g =$ right(e) for some edge e of path $\pi\}$

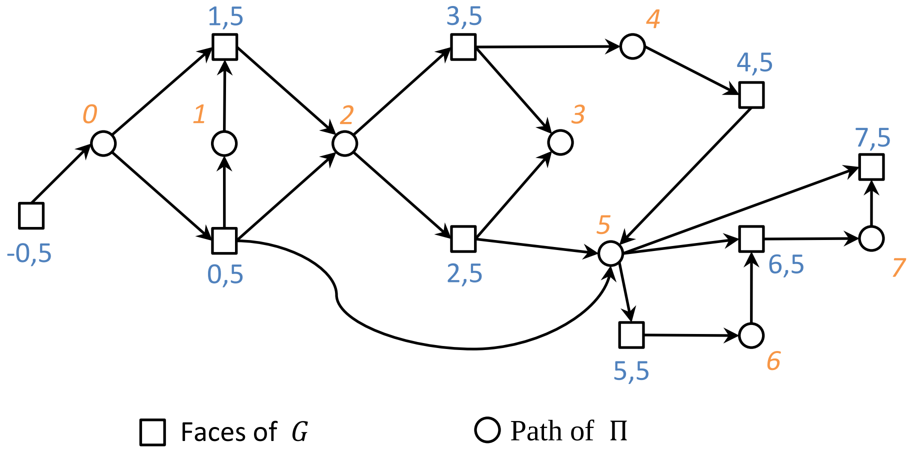
2. Assign unit weights to the edges of G and compute an optimal topological numbering. Y of G with $Y(s) = 0$.
3. Assign half-unit weights to the edges of G_Π and compute an optimal topological numbering X of G_Π such that $X(s^*) = -\frac{1}{2}$.
4. For each path $\pi \in \Pi$ do:

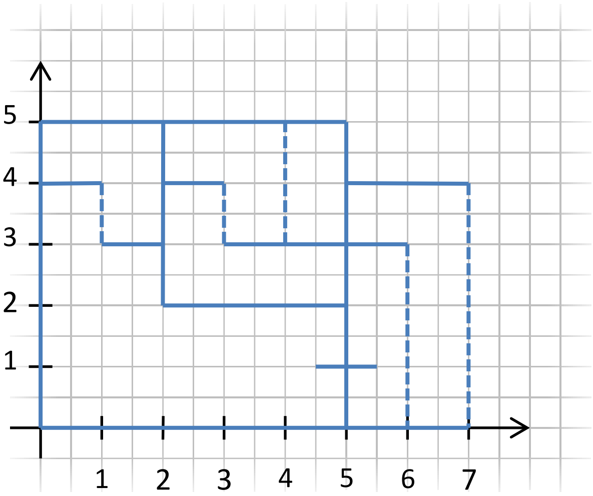
```
For each edge $e \in \pi$ do:
    draw $\Gamma(e)$ as vertical segment:
    $x(\Gamma(e)) = X(\pi)$
    $y_B(\Gamma(e)) = Y(\text{orig}(e))$
    $y_T(\Gamma(e)) = Y(\text{dest}(e))$
```

5. For each vertex $v \in G$ do:

```
draw $\Gamma(v)$ as horizontal segment with:
$y(\Gamma(v)) = Y(v)$
$x_L(\Gamma(v)) = \min_{v\in\pi} X(\pi)$
$x_R(\Gamma(v)) = \max_{v\in\pi} X(\pi)$
```





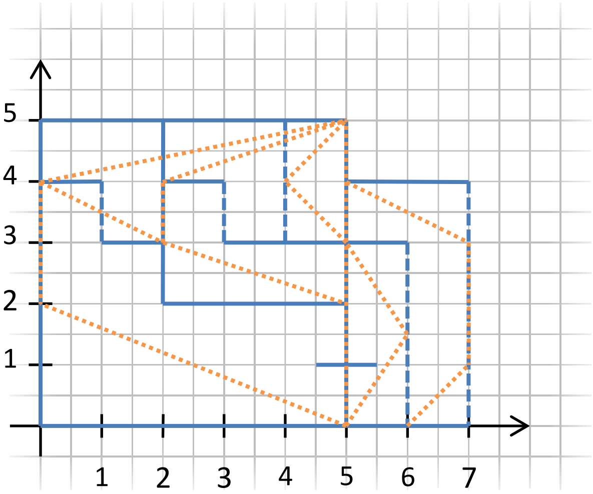


Lemma
The digraph G_{Π} constructed in step 1 of Algorithm 3 is a planar st-graph.

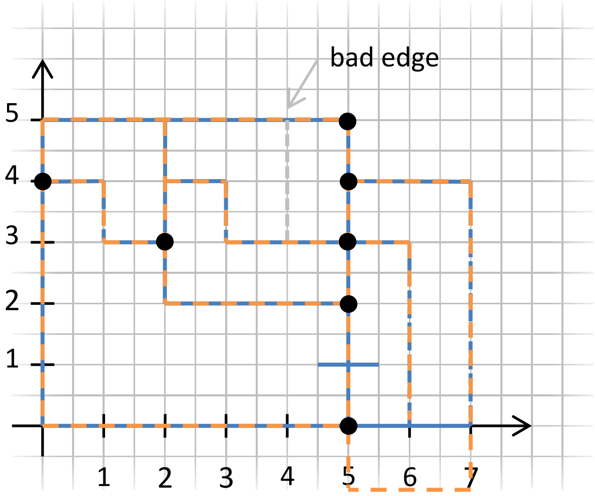
Theorem
Let G be a planar st-graph with n vertices and let Π be a set of nonintersecting paths, covering the edges of G. Algorithm 3 computes in $O(n)$ a visibility representation of G with integer coordinates and $O(n^2)$ area such that the edge of every path $\pi \in \Pi$ is vertically aligned.

From a constrained visibility representation we can construct:

- Constrained polyline drawing in $O(n)$ time.
    1. Vertex segment → point
       IF $v \in \Pi$:$x(u) = X(\pi)$; $y(v) = Y(\Pi)$
       otherwise free to chose

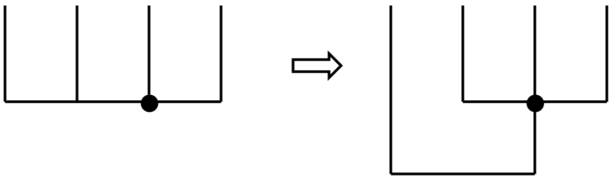


2. Edge segment → edge
   IF $y(v) - y(u) = 1$:
   segment with ends in $P(u), P(v)$
   else:
   polygonal line from $P(u)$ to $P(v)$
   through the points
   $\left(x\left(\tau(u,v)\right), y(u) + 1\right)$;
   $\left(x\left(\tau(u,v)\right), y(v) - 1\right)$



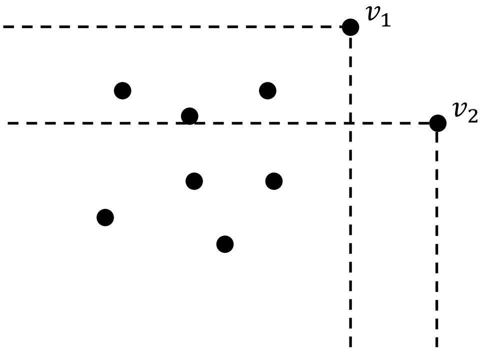
- Planar orthogonal drawing in $O(n)$ time
  assume max degree ≤ 4
  1. Vertices:
     s/t draw at intersection of vertex segment with median edges.
     All other vertices:
     Drawn at intersection $P(v)$ of vertex segment with the edge segments of path π_v
  2. Edges:
     $e = (u,v)(u,v \notin \{s/t\})$, drawn as orthogonal chain trough points $P(u)$; intersection of $\tau(u)$ and $\tau(e)$; intersection of $\tau(e)$and $\tau(v)$; and $P(v)$.
     ⇒ 3 segments (first and last may be empty)

Edges incident to s, t:

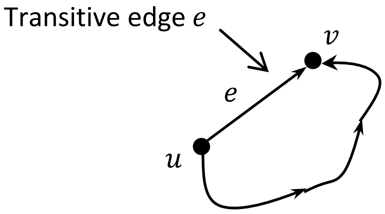


2.3 Dominance drawings

DEFINITION
A dominance drawing τ of a digraph G, is drawing such that, for any two vertices u, v there is a directed path from $u \to v$ in G iff $x(u) \leq x(v)$ and $y(u) \leq y(v)$ in τ. "v is dominating u"



DEFINITION
A digraph without transitive edges is called "reduced".

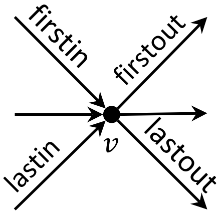


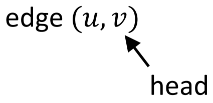
LEMMA
Any strait-line dominance drawing Γ of a reduced planar st-graph G is planar.

ALGORITHM 4 – DOMINANCE STRAIGHT-LINE
See handout.

3 PHASES

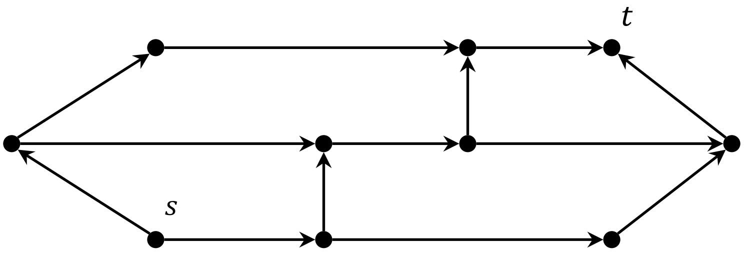
- Preprocessing → set up datastructure



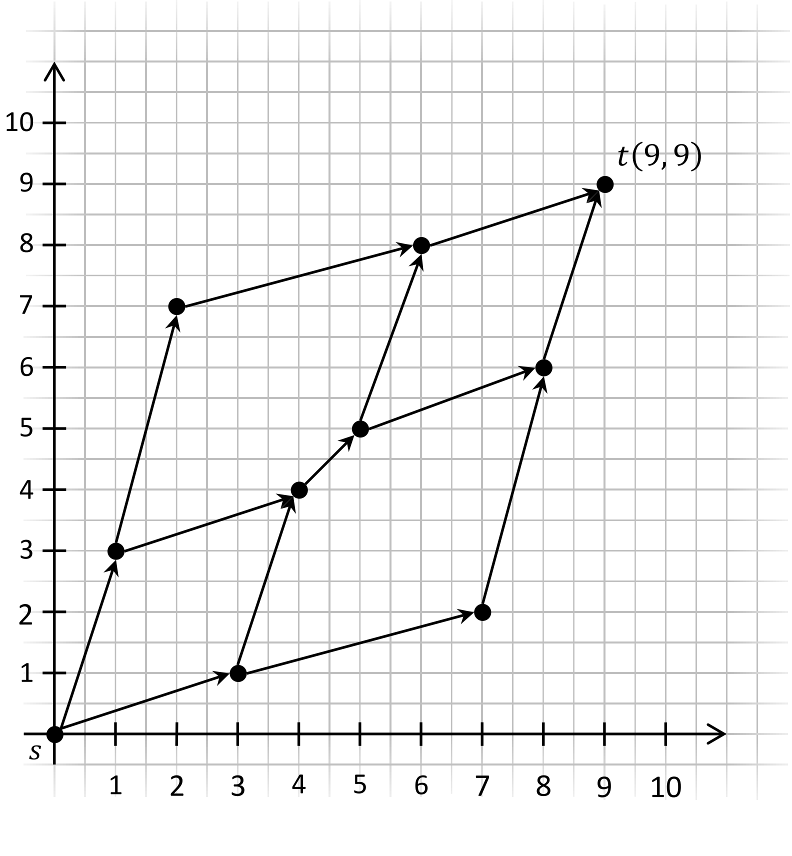


- Preliminary layout → Assign to all vertices distinct (x, y) coordinates.
  Essentially: 2 topological sortings; scanning for each vertex the successers; clockwise and counterclockwise
- Compaction → Reduce area of final drawing

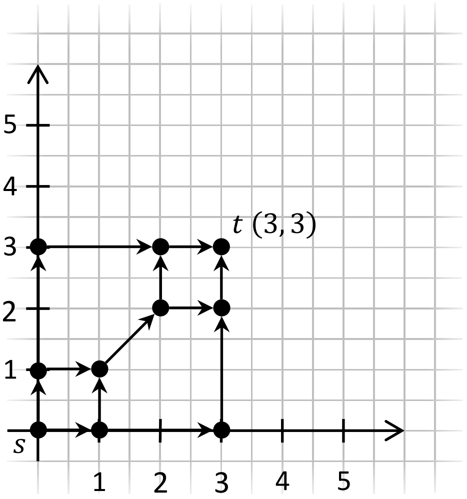
EXAMPLE



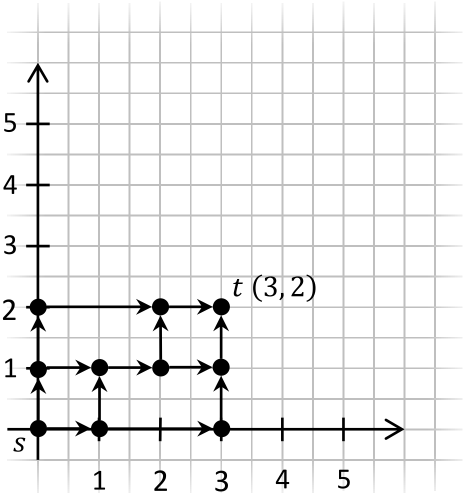
Preliminary layout:



After compaction:



Minimal drawing:

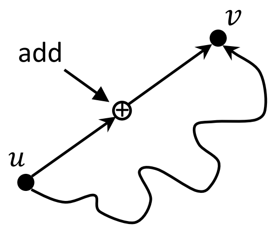


THEOREM
Let G be a reduced planar st-graph with n vertices. Algorithm 4 dominance straight line constructs in $O(n)$ time a planar straight-line dominance grid drawing Γ of G with $O(n^2)$ area.

- Symmetries
  Any components of G that are isomorphic (same direction of edges + boundaries of the faces) or symmetric (axially or rotationally isomorphic to themselves) have straight-line drawings that are congruent (up to a translation-reflection) or symmetric (with respect to the line through source/sink, or 180° rotation through its centroid)
- Minimum area drawings

Algorithm can be modified but symmetries may be lost

- General planar st-graphs
  Add dummy vertex on all transitive edges



3.1 Orthogonal drawings with maximum-degree 4

Definition

An st-numbering for a graph G with n vertices is a numbering for $s = v_1, v_2, \dots, v_n = t$ of the vertices of G such that every vertex $v_j \notin \{s, t\}$ is adjacent to at least two vertices v_i, v_k with $i < j < k$.

We orient every edge of G from the low number vertex to the high numbered one.

Basic Algorithm idea

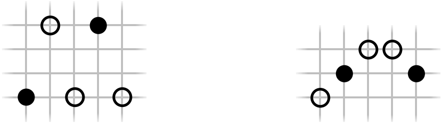
- Compute st-numbering
- Place first vertex on grid. Allocate columns for all incident edges
- Place other vertices on grid according to st-numbering:
  - Vertex v gets a new row
  - All incoming edges of v are drawn on the already allocated columns
  - All outgoing edges of v are "drawn" by allocating one new column per edge

⇒

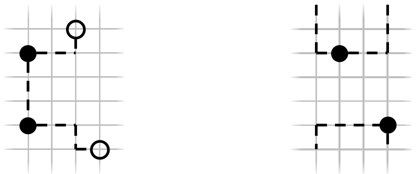
- $O(n)$ time
- $O(n^2)$ area
- At most $2n + 4$ bends (at most 2 per edge)

Pair vertices for space reduction:

1. Row pairs:
   a. Both vertices share the same row
   b. Vertices of such a pair are placed in different rows, but their placement results in reusing one row

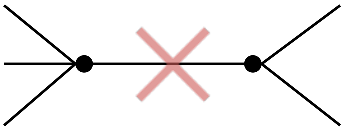


2. Column pairs
   a. The two vertices are placed such that at least two different edges use the same column



Definition
A vertex with a incoming and b outgoing edges is a a-b vertex $(0 \leq a + b \leq 4)$

Condense G
Scan G for 1-1 vertices whose outgoing edge enters a 1-2 vertex or a 1-3 vertex. Absorb these 1-1 vertices into a single edge until no 1-2 or 1-3 vertex has a 1-1 vertex as immediate predecessor.



Condensed graph $G' |V(G')| = n'$. Need to modify the st-numbering to remove gaps.

Definition

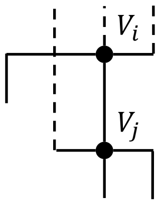
- An assigned vertex belongs to a pair; an unassigned vertex does not
- The next unassigned vertex considered is always a $\boxed{\text{1-2, 1-3, 2-2}}$ vertex and it is paired with some lower numbered vertex in G'
- The vertex of a pair with the lower st-numbering is called the first vertex, the other is the second vertex
- A predecessor of a vertex in G or G' with respect to the st-numbering is the

immediate predecessor of the vertex

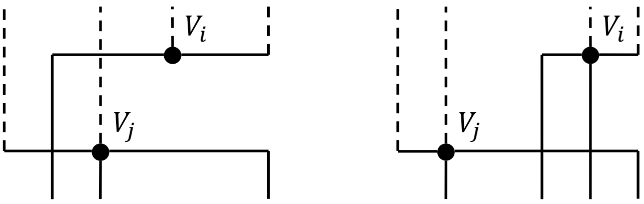
- Let (v_i, v_j) $(j < i)$ be a pair formed by Algorithm 5. If they are not predecessor-successor they are called <u>independent</u>

Case analysis for drawing pairs

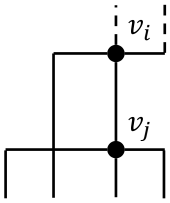
1) v_i is a 2-2 vertex
    a. v_j is a 2-2 vertex and predecessor of v_i



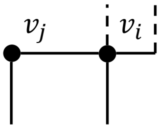
    b. v_j is a 2-2 vertex independent of v_i



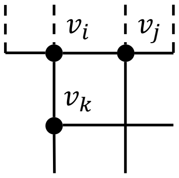
    c. v_j is a 2-1 or 3-1 vertex and predecessor of v_i



    d. v_j is a 1-1 vertex and predecessor of v_i

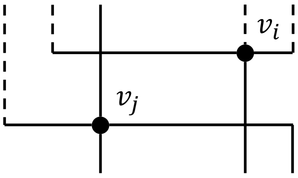


    e. v_j is a 1-2 or 1-3 vertex and predecessor of v_i



$k < j$
v_k other predecessor of

f. v_j is a 1-2 or 1-3 vertex and independent of v_i

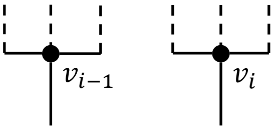


2) v_i is a 1-2 or 1-3 vertex
→ it is always paired with v_{i-1}

a. v_{i-1} is a 2-2, 2-1 or 3-1 vertex

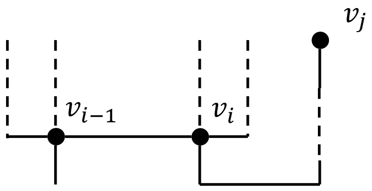
Reuse column as in cases 1a – 1c

b. v_{i-1} is a 1-2, 1-3 vertex independent from v_i

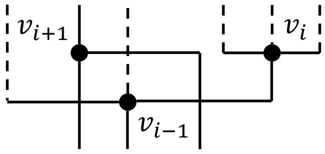


c. v_{i-1} is a 1-2 or 1-3vertex predecessor of v_i
→ We have a row pair if both of the following conditions hold:

- The edge (v_{i-1}, v_i) has not absorbed any 1-1 vertices of G
- Either v_i is connected later to another vertex v_j which is 1-1, 1-2, 1-3 OR v_i is connected later to a 2-2 vertex, which is the second vertex of a pair of type 1.d) OR 1.e)



e. v_{i-1} id a 1-2 or 1-3 vertex and at least one of the conditions of 2.c) does not hold



ALGORITHM 6 – STEP 6

Place absorbed vertices on bends or grid points where are no crossings. If not possible: introduce new rows.

LEMMA

- Suppose there is a total of $p_1$column pairs, p_2 unassigned degree 2 vertices, p_3 unassigned degree 3 vertices in G:
  Let $k_1 = p_1 + p_2 + \frac{p_3}{2}$ then Algorithm 6 produces a drawing of width at most $n + 1 - k_1$
- Suppose that k_2 is the number of row pairs in G then Algorithmen 6 produces a drawing of height at most $n + 1 - k_2$ if $k_2 \geq 1$ OR n if $k_2 = 0$

THEOREM

Let G be a biconnected graph G with n vertices and maximum degree 4. Algorithm 6 Four-Orthogonal constructs an orthogonal grid drawing Γ of G in linear time $O(n)$ such that:

- Γ has area at most $0.77n^2 + O(n)$
- Γ has a total number of $2n + 4$ bends
- No edge of Γ has more than 2 bends

RUNNING TIME

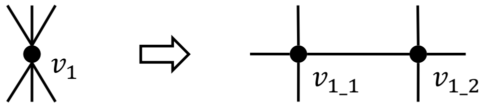
- Insert rows on top of existing drawing → easy
- Insert columns anywhere → problem

Solution: Order maintenance problem

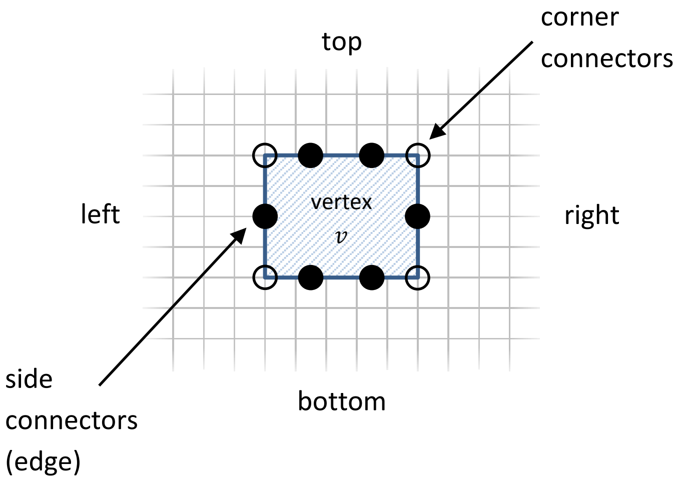
Insert(x, y): Insert y directly after x
Delete(x):
Order(x, y): True if x is before y in the list.
False if not

All operations: $O(1)$ time

3.2 Orthogonal drawings with maximum-degree > 4



Better: Each vertex is drawn as rectangular box with 4 sides lying on the grid. Each side (except bottom) has a number of connections with integer coordinates where incident edges are attached.



ALGORITHM

- Insert vertices in st-ordering
- Decide size/exact placement of vertex when inserting

SIZE

- All outgoing edges connect to the top of the box
- Incoming edges are split between the left side $\left\lceil\frac{\text{indeg}(v)}{2}\right\rceil$ and the right side $\left\lfloor\frac{\text{indeg}(v)}{2}\right\rfloor$ of the box

Position

Columns:

- Order all columns of the current drawing from left to right (including all columns with incoming edges) compute the columns C_1, C_2 containing the median incoming edges of v.
- Insert needed amount of columns for v between C_1 and C_2 (one median C: right c)

Rows:

- Insert needed amount on top of the existing drawing.

Final

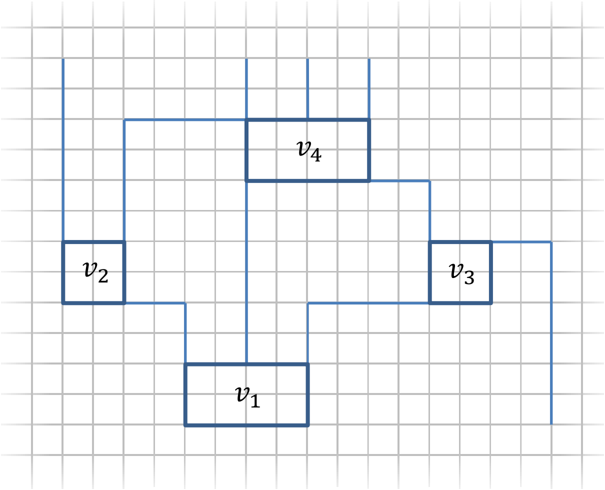
- Put box in the newly created space
- Connect all incoming edges from column C_1 and left to the left side; all incoming edges from column C_2 and right to the right side of the box

Given a graph with m edges and st-ordering

$\Rightarrow$ width $\leq m + \underset{\nwarrow}{n_{out1}}$ number of vertices with outdegree = 1

height $\leq \frac{m}{2} + \frac{n_{odd}}{2} + n_{in1} + n_{in2}$

$\nwarrow$ number of vertices with odd degree



Improve

Rows/columns may be shared. Vertices of degree 2 and some of degree 3, 4 can be represented as points.

Theorem

Let G be a graph with m edges and an st-numbering. There exists an Algorithm that produces an orthogonal gridding Γ of G, by representing the height degree vertices as boxes. Γ can be computed in $O(m)$ time and it has the properties:

- The perimeter of each vertex is proportional to its degree
- Width of Γ is at most $m - 1$
- Height of Γ is at most $\frac{m}{2} - 2$
- Each edge has at most one bend
- Total number of bends is at most $m - n_{out1}$

number of vertices
with out-degree = 1

4.1 Layered drawings of digraphs

Given: Digraph G

Wanted: Layered polyline drawing Γ of G

Steps:

0. *Cycle Removal* (Preprocessing)
   Temporarily reverse edge directions to make G acyclic

1. *Layer assignment*
   Assign vertices to horizontal layers (y- coordinates)

2. *Crossing Reduction*
   Order vertices within layers to reduce crossings

3. *Assign x-coordinates*

4. *If necessary put cycles back in*

4.1.1 Layer assignment

Definition

Suppose that $G = (V, E)$ is a cyclic digraph:

- A layering of G is a partition of V into subsets $L_1, L_2, \dots, L_n$ such that:
  If $(u, v) \in E$ where $u \in L_i, v \in L_j$, then $i > j$
- The height of such a layering is the number of layers h
- G is then called a (h-)layered digraph
- The width of a layered digraph is the

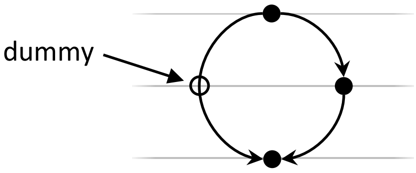
number of vertices in the largest layer

$$\max_{1 \le i \le h} |L_i|$$

- The span of an edge $(u, v)$with $u \in L_i$, $v \in L_j$ is $i - j$
- In a proper digraph, all edges have span = 1

Requirements of layering

- Compactness (small width/height)
- Proper (if not insert dummy vertices on “long” edges)
- Small number of dummy vertices
  - Running times depend on all vertices
  - Bends occur only at dummy vertices



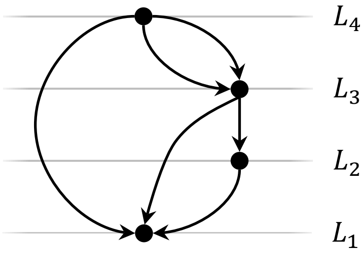
Minimum height layering

Algorithm 7 - Longest path layering

Input: Reduced, acyclic graph G

Output: Layering of G with minimum height

1. Place all sinks in L_1
2. Each remaining vertex v is placed in layer L_{p+1} where the longest path from $v \rightarrow$ sink has length p



Properties of the drawing produces b Algorithm 7:

J Linear time

J Minimum number of layers

L Drawings may be too wide

Minimum width layering

Finding a layering with minimum height and minimum width → NP complete

⇒ Heuristic that gives a layering with width at

most w and height at most $h \le \left(2 - \frac{2}{w}\right) h_{\min}$.
$h_{\min}$ minimum height of layering of width w.

Note

Width of layering does not take dummy vertices into account!

Algorithm 8 – Coffman-Graham layering

Input: Reduced digraph $G = (V, E)$ and $w \in \mathbb{N}$
Output: Layering of G with width w

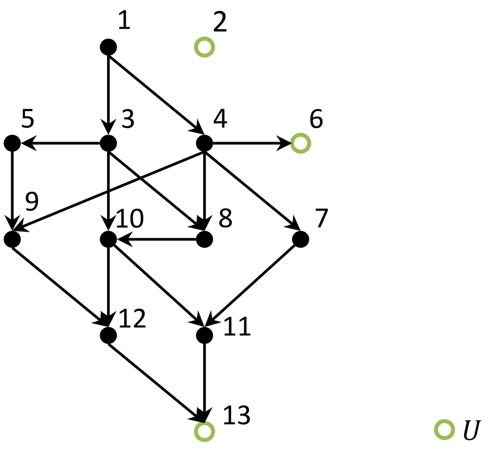
```
1. Initially, all vertices are unlabeled
2. For $i = 1$ to $|v|$ do
      a. Choose unlabeled vertex $v$ such that
         $\{\pi(u) | u, v \in E\}$ is minimized
      b. $\pi(v) = i$

3. $k = 1, L_1 = \emptyset, U = \emptyset$
4. While $U \neq V$ do
      a. Choose $u \in V - U$ such that every
         vertex in $\{v | (u, v) \in E\}$ is in $U$ and
         $\pi(u)$ is maximized
      b. If $|L_k| < w$ and for every edge $(u, v)$,
         $x \in L_1 \cup \ldots \cup L_{k-1}$ then add $u$ to $L_k$
         else $k = k + 1$, $\mathbb{L}_k = \{u\}$
      c. Add $u$ to $U$
```

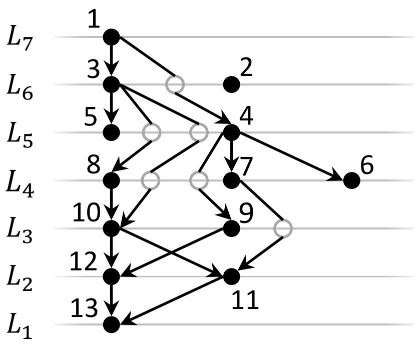
Details to Algorithm 8 – Step 2

Algorithm 8 uses lexicographic order defined on finite sets of integer > 0. The largest item of the set is the most significant!

$$\{1,4,7\} < \{3, \mathbf{8}\}$$
$$\{1,5, \mathbf{8}, 9\} > \{3,4,9\}$$

Example





○ dummy vertices

$w = 3$

Minimize number of dummy vertices
Suppose each vertex u has a y-coordinate $y(u)$ with the properties:

1. $y(u)$ is an integer $\forall u$
2. $y(u) \geq 1 \quad \forall u$
3. $y(u) - y(v) \geq 1$ for each edge (u, v)

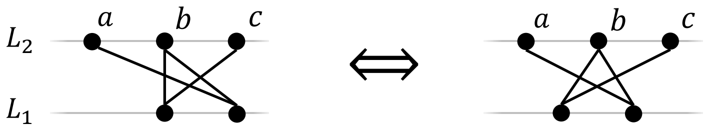
The function y defines a layering with $L_m = \{u \in V | y(u) = m\}$. Let $f = \sum_{(u,v)\in E}(y(u) - y(v) - 1)$ be the sum of the vertical spans of the edges in this layering minus the number of edges.

The layer assignment problem is reduced to choosing y-coordinates to minimize f subject to conditions 1-3 integer linear problem
⇒ Polynomially solvable

4.1.2 Crossing reduction

Observation

The number of edge crossings in a layered digraph does depend only on the ordering of the vertices.



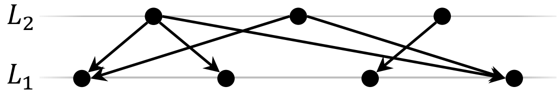
⇒ Combinatorial problem!

Minimize edge crossings in a layered digraph
⇒ NP complete

Layer-by-layer sweep

Definition

A two layered digraph is a bipartite digraph $G = \{L_1, L_2, E)$ consisting of disjoint vertex sets L_1, L_2 and a set $E \subseteq L_1, L_2$ of edges.



We specify vertex orderings by giving unique x-coordinates $x_i(u)$ to each vertex $u \in L_1; i = 1,2$

Definition

- The number of crossings in a digraph of G specified by x_1, x_2 is $\text{cross}(G, x_1, x_2)$
- The minimum number of crossings, when the vertices of L_1 are ordered by x_1 is $\text{opt}(G, x_1)$

It holds: $opt(G, x_1) = \min_{x_2} \text{cross}(G, x_1, x_2)$

Definition

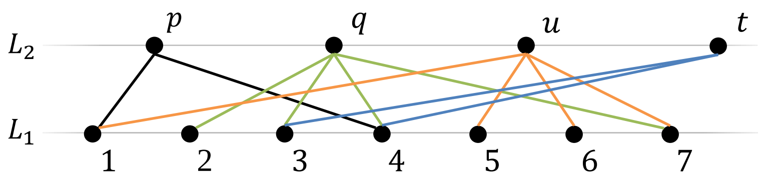
We define the two-layer crossing problem as follows. Given a two layered graph $G = (L_1, L_2, E)$ and an ordering x_1 on L_1 find an ordering x_2 on L_2 such that $\text{cross}(G, x_1, x_2) = \text{opt}(G, x_1)$

⇒ NP-Complete

Definition

Let u, v be distinct vertices in L_2. The crossing number C_{uv} (for $u \neq v \in L_2$) is the number of pairs $(u, w), (v, z)$ of edges with $x_1(w) > x_2(z)$ & $x_2(u) < x_2(v)$ & $C_{uv} = 0$.

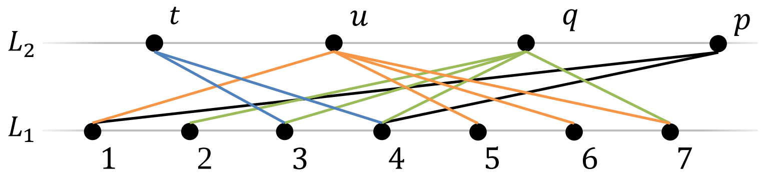
Example



	p	q	u	t
p	0	2	1	1
q	5	0	6	3
u	6	9	0	6
t	2	3	2	0

order of vertices is p, q, u, t

order of vertices is t, u, q, p



Lemma

If $G = (L_1, L_2, E)$ is a 2-layer digraph and x_1, x_2 are orderings of L_1 and L_2 then

$$\operatorname{cross}(G, x_1, x_2) = \sum\nolimits_{x_2(u) < x_2(v)} C_{uv}$$

Further:

$$\operatorname{opt}(G, x_1) \geq \sum\nolimits_{u,v} \min(C_{uv}, C_{vu})$$

Where the sum is over all unordered pairs $\{u, v\}$ of vertices in L_2.

Sorting methods

- Adjacent exchange
  - Scan the vertices of L_2 from left to right, exchanging all adjacent vertices u, v if $C_{uv} > C_{vu}$; until the number of crossings does not reduce ("Bubble Sort")

- Split
  - Choose pivot vertex $p \in L_2$
  - For each vertex $u \in L_2 \quad u \neq p$ do:

    If $C_{up} < C_{pu}$

    Then place $u \in V_{\text{left}}$

    Else place $u \in V_{\text{right}}$
  - Recurs on $V_{\text{left/right}}$ concatenate ("Quicksort")

Problem with sorting methods: Both need pre compute of crossings array!

AVERAGING METHOD

- Barycenter Method

  x_2 is the barycenter for all $u \in L_2$.

  $$\operatorname{avg}(u) = \frac{1}{\deg(u)} \sum_{v \in N_u} x_1(v)$$

  neighbors of u

  If two vertices have the same barycenter, we separate them arbitrarily by a small amount. The number of crossings achieved is $\operatorname{avg}(G, x_1)$

- Median method

  If the neighbors of $u \in L_2$ are $v_1, v_2, \dots, v_j$ with $x_1(v_1) < x_1(v_2) < \cdots < x_1(x_j)$ then we define $\operatorname{med}(u) = x_1\left(v_{\lfloor j/2 \rfloor}\right)$.

  If u has no neighbors, $\operatorname{med}(u) = 0$.

  We order the vertices of L_2 by sorting them on $\operatorname{med}(u)$.

  If $\operatorname{med}(u) = \operatorname{med}(v)$, and say $\deg(v)$ is odd and $\deg(u)$ is even, then $x_2(v) < x_2(u)$.

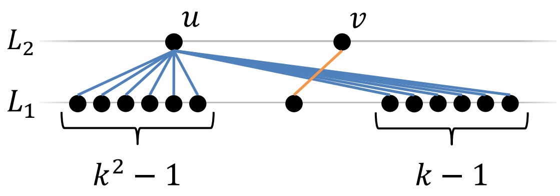
  If both degrees are odd or even, we place u and v arbitrarily.

  The number of crossings achieved is $\operatorname{med}(G, x_1)$

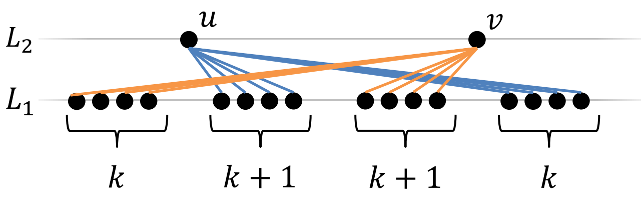
THEOREM

Suppose that $G = (L_1, L_2, E)$ is a two-layer digraph and x_1 is an ordering of L_1, if $\operatorname{opt}(G, x_1) = 0$ then $\operatorname{avg}(G, x_1) = \operatorname{med}(G, x_1) = 0$. Neither method gives an optimal solution in all cases!

Worst case barycenter



Worst case median



Application-hybrid approach

1. Initial ordering with median method
2. Tiebreak with barycenter method
3. Define the output with adjacent exchange

4.1.3 Horizontal coordinate assignment

Consider the directed path $p = (v_1, v_2, \dots, v_{k-1}, v_k)$ where $v_2, \dots, v_{k-1}$ are dummy vertices. If this path was drawn straight, then the x-coordinates of the dummy vertices v would satisfy:

$$x(v_i) - x(v_1) - \frac{i-1}{k-1}\big(x(v_k) - x(v_1)\big)$$

For each of these path p corresponding to an edge with span > 1, we define

$$g(p) = \sum_{i=2}^{k-1}\left(x(v_i) - \frac{i-1}{k-1}\big(x(v_k) - x(v_1)\big) + x(v_1)\right)^2$$

To make the edges as trait as possible, we minimize the global sum $\sum g(p)$ over all paths of dummy vertices subject to the constraints $x(w) - x(z) \geq \delta$ for all pairs w, z of vertices in the same layer (w right of z).

Constraints ensure:

- Ordering is maintained
- Minimal horizontal distance of δ between vertices

Add further constraints:

- X-coordinates lie within width boundary

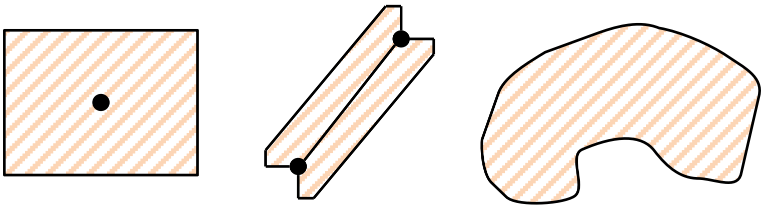
5.1 Label placement

Assume

Drawing is fixed, may not be changed.

Definition

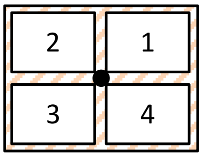
A label is a textural description that conveys information or clarifies the meaning of complex structures presented in graphical form.
Let Γ be a drawing and F the set of features to be labeled. A solution to the labeling problem for Γ assigns labels to each feature $f \in F$ such that the relevant information is communicated "in the best possible way". This can be achieved by positioning the labels "in the most appropriate place". For each feature there is a large number of potential label positions, the labeling space.



Label quality evaluation

Basic rules:

1. No overlap of a label with other labels or features
2. Each label can be easily identified with exactly one feature in the drawing
3. Each label must be placed in the best possible position (among all acceptable ones)



Definition

Given a set F of graphical features to be labeled, we define:

Λ is the set of all label positions

Λ_f the set of all label positions for feature $f \in F$

λ $F \to \Lambda$ a function assigning a label position to a feature in example $\Lambda(f) = \lambda_f \in \Lambda_f$

OPTIMIZATION PROBLEM

Each label position is associated with a cost function: $\Lambda \to \mathbb{N}$. Cost is determined with respect to the quality of a label position. We want to find a label assignment to all features that minimizes the total cost.

THE LABELING PROBLEM

Given: Set of features F

Find: Label assignment minimizing
$\sum_{i \in F} \sum_{j \in \Lambda} \mathrm{cost}\big(\lambda(i)\big) * P(i,j)$ where
$$P(i,j) = \begin{cases} 1 & \lambda(i) = j \\ 0 & \text{otherwise} \end{cases}$$
And $\sum_{i \in F} \sum_{j \in \Lambda} P(i,j) = |F|$ where
$\sum_{j \in \Lambda_i} P(i,j) = 1 \quad i \in F$

5.2 Graphical feature label placement

Most general method!

ALGORITHM 9 – BASIC LABELING

Input: A drawing Γ and a set of objects F to be labeled.

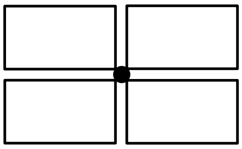
Output: A label assignment free of overlaps

1. Select label positions for each object
2. Remove heavily overlapping labels; Group overlapping labels together
3. Final label assignment: Solve a matching problem

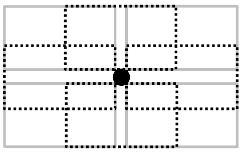
HEURISTICS TO SELECT LABEL POSITIONS

Points: Label positions touching the corresponding point.

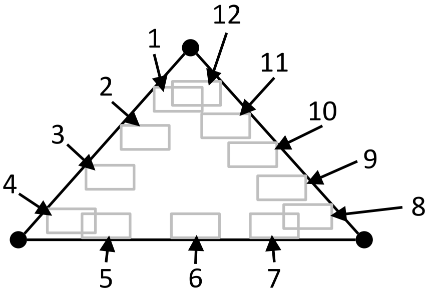
4:



8:



Edges: Define a number of equally spaced points on the edge. Each label position is associated with exactly one of those points and touches it in a corner:



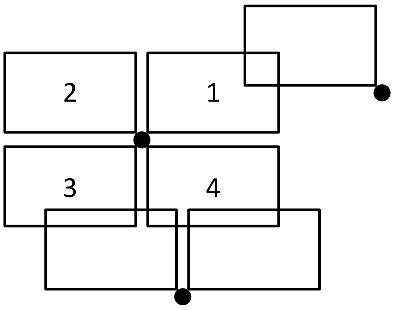
Reducing the number of labels

Create an intersecting graph: Each label position is a node, if 2 positions intersect, we add an edge. Finding intersections between n labeling rectangles: $O(n \log n + k)$

Goal: Reduce the intersection graph to a set of disconnected subgraphs.

Preprocessing: Remove all label positions that add to the complexity of the problem without (potentially) improving the solution.

Example



⇒ Remove 3 & 4.
Keep 1 for later possible use

Main step

If a subgraph C must be split, we remove the node with highest degree.
Unless it corresponds to a label position of some object with very few label positions. → Choose second highest degree.
Repeat until C is split in two disjoint subgraphs or C is a complete graph.

Matching labels to objects

Definition

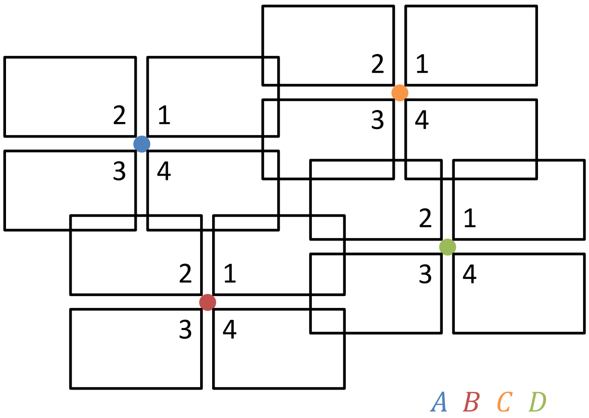
Given a drawing Γ, a set F of graphical features to be labeled, and a set Λ of label positions for

F. Then we define the matching graph $G_m(V_f, V_C, E_m)$:

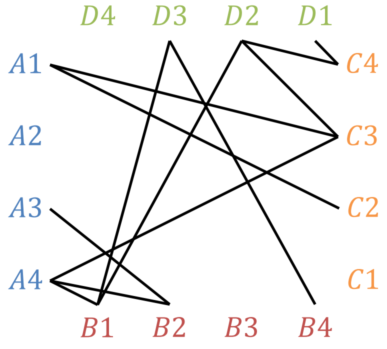
- Each node $f \in V_f$ corresponds to a feature $f \in F$
- Each node $c \in V_C$ corresponds to a group of overlapping labels
- Each edge $(f, c) \in E_m$ connects a node $f \in V_f$ to a node $c \in V_c$ iff $f \in F$ has a label position that is a member of group C
- G_m is bipartite!
- The cost of assigning label l to feature f is the weight of edge $(f, l) \in E_m$
- The size of G_m depends
    - On the size of the input drawing (V_f)
    - On the size of the set of labels (V_C)
    - On the density of the overlaps (E_m)

A final label assignment can be found by computing the maximum cardinality minimum weight matching on G_m. This can be done in $O(n^{2.376})$ randomized time. Or in $O\left(\sqrt{\underset{\substack{\nearrow\\ \#\text{nodes}}}{n}} * \underset{\substack{\uparrow\\ \#\text{edges}}}{m}\right)$ deterministic time.

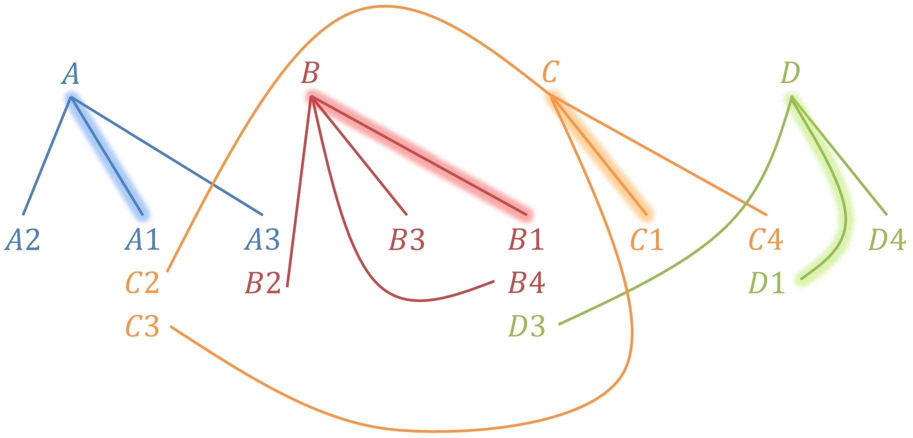
Example



Create the intersection graph:

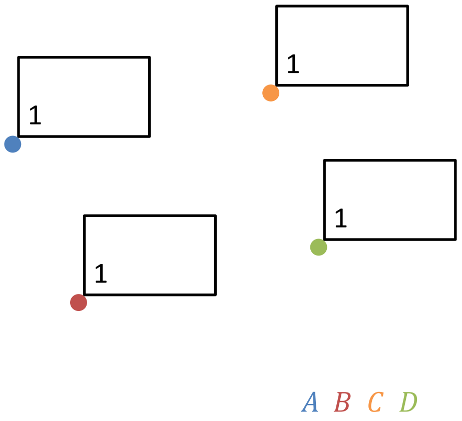


Simplify! Remove all nodes with degree ≥ 3
⇒ $A4$ & $D2$
Create the matching graph:



Find maximum cardinality minimum weight matching on G_n

SOLUTION



5.3 Edge label placement (ELP)

Assumption

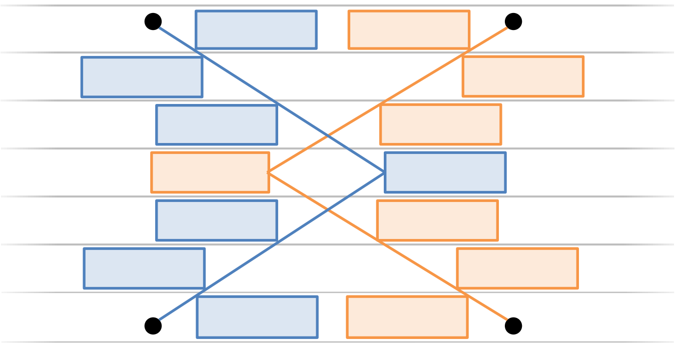
1. All labels have the same height
2. Each edge has only one label assigned to it

Wanted

Assign to each edge a label position free of overlaps, that touches only "its" edge. Finding the initial set of label positions.

- Divide the input drawing into consecutive horizontal strips of equal height ≙ height of labels
- Each label must lie fully inside one horizontal strip. → "Slide" the label inside the strip until it touches the correct edge.
- A label is only included in Λ_e if it does not overlap any other graphical features. Overlapping other label positions is allowed.

Example



Observations

- A label position of edge e does not overlap any other label position of e
- If to label positions overlap, they must lie inside the same horizontal strip
- Each label position overlaps at most <u>one</u> label position

Algorithm 10 – Edge labeling

Input: A drawing Γ of graph $G = (V, E)$

Output: A label assignment to the edge without overlap

1. Split Γ into horizontal strips of label heights

2. Find all label positions for each edge and construct the groups of overlapping labels
3. Construct the matching graph G_m of Γ
4. Match label positions to edges:
    - Find maximal cardinality
    - Minimum weight matching of G_m

In G_m each v_c has degree at most 2.
→ Matching heuristic in linear time

ALGORITHM 11 – FAST MATCHING

Input: Matching Graph G_m

Output: A maximum cardinality matching for G_m with low total weight.

1. If the minimum weight incident edge of $f \in V_f$ connects to $c \in V_c$ with $\text{degree} = 1$, then:
    a. Assign this edge as matched edge
    b. Update G_m:
        - Remove f, c and all incident edges
        - Store $f, c, (f, c)$ as part of the matching
2. If a node $f \in V_f$ has $\text{degree} = 1$ then:
    a. Assign its incident edge as matched edge
    b. Update G_m
3. Repeat 1. & 2. Until no more edge can be matched
4. Delete all nodes with $\text{degree} = 0$ from G_m
5. For each node $f \in V_f$ do:
    Remove all edges except the two with lowest weight
6. G_m consists of simple path and cycles
    a. Find the 2 possible maximum cardinality matchings for each path/cycle
    b. Choose the matching with minimum weight

☺ Easy!
☺ Works well if the edges are long and vertical → Many potential label positions! Good for hierarchical and strait line drawings

☹ Cannot deal with horizontal edges!
→ Bad for orthogonal drawings

→ Solution: Add vertical strips

☹ Some edges may get no label or outcome is not satisfactory.
→ Solution: Manual tweaking

5.4 Label Placement by drawing modification

If we do not have a graphic or technical map, we can modify the drawing to help with label placement.

1. Modify the existing layout to make room from the labels. (Open problem!)
2. Produce a new drawing integrating the layout and the labeling process. → For orthogonal drawings

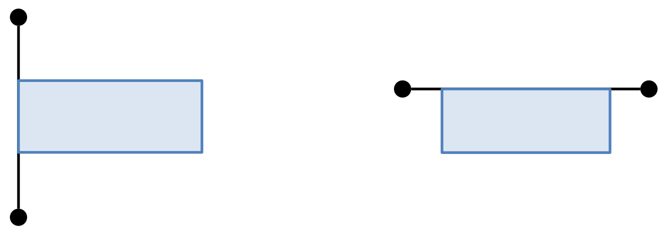
To No. 2

Given: Let G be a planaer graph with orthogonal representation H. Let L be a set of labels for the edges of G (1 label per edge) modeled as axis parallel rectangle with given height and width.

Want: A orthogonal grid drawing of G such that the edges are labeled and have the shape defined in H.

CONSTRAINTS AND GOALS FOR A "GOOD" DRAWING

1. A label is drawn with one of its sides as a proper subset of its edge.



2. A label of segment s cannot overlap any other label, vertex or segment
3. A good label placement minimizes the area of the drawing. This problem can be formulated as integer linear program and solved by heuristics (Branch & Cut)

6.1 Rectangular cartograms[1]

Definition

A cartogram is a map where the size of a region is not the true size, but corresponds to a particular geographic variable (e.g. population). The geometry is usually distorted to convey the information of the alternate variable. “Value-by-area” map.

cartogram	↔	Kartenanamorphote
thematic map	↔	Kartogramm

Generals types of cartograms

- Contiguous area cartograms
  Have deformed regions so that the desired size can be obtained and the adjacencies kept. (standard model)
- Non-contiguous area cartograms
  The region have their true shape, but are sealed down and generally do not touch anymore
- Cartograms based on circles
- Rectangular cartograms
  Each region is represented by a single rectangle
- Hybrid cartograms
  Hybrid of the first and forth type: Regions are rectilinear polygons with small number of vertices

Rectangular cartograms (1934)

☺ Areas can be computed easily

☹ Rectangular shape is less recognizable and imposes limitations to the possible layout

Quality criteria

- Cartographic error for each region

  $$\frac{|A_C - A_S|}{A_S}$$

  A_c =area of the region in the cartogram
  A_s =area specified by the variable to be shown
- Average and maximum cartographic error
- Correct adjacencies of the rectangles
- Maximum aspect ratio
- Suitable relative positions

[1] **On Rectangular Cartograms**, M. van Kreveld & B. Speckmann, www.cs.uu.nl/research/techreps/repo/CS-2004/2004-040.pdf

DEFINITION

- A rectangular partition of a rectangle R is a partition of R into a set S of non-overlapping rectangles such that no 4 rectangles in S meet in the same point.
- A rectangular dual of a plane graph G is a rectangular partition R such that:
  1. There is a one-to-one correspondence between the rectangles in R and the node in G
  2. Two rectangles in R share a common boundary iff corresponding nodes in G are connected
- A triangle is a cycle of G consisting of 3 areas
- A cycle C of G divides the plane into an interior and an exterior region. If C contains at least one vertex in its interior and exterior then C is a separation cycle

THEOREM

A plane graph G has a rectangular dual R with 4 rectangles on the boundary of $\mathcal{R}$ if and only if

1. Every interior face is a triangle and the exterior face is a quadrangle
2. G has no separating triangles

NOTE

- An error-free cartogram need not exist even if a rectangular dual does
- A rectangular dual need not be unique
- Planar graphs can always be represented with rectangles, L-shapes and T-shapes

6.2 Algorithm 12 – Rectangular dual[2]

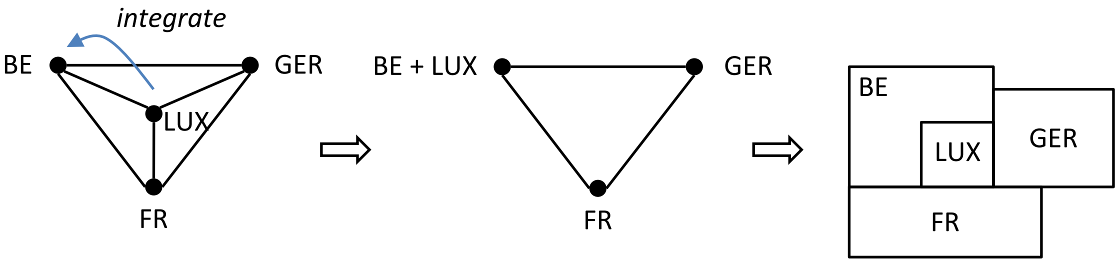
DEFINITION

A face graph F of a map has a vertex for each region in the map and an edge between 2 vertices iff their regions share a common boundary

1. Preprocessing
   - Construct the face graph of the map
   - Triangulate all non-triangular faces of E
     → Each triangulation leads to a different rectangular dual!

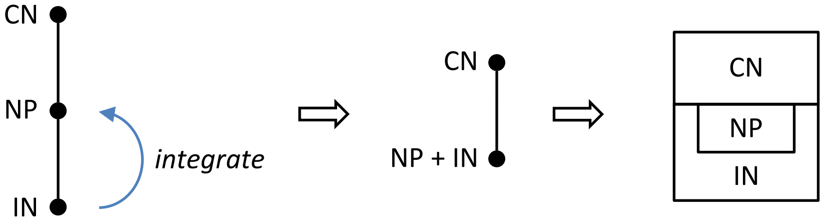
[2] **Regular edge labeling of 4-connected plane graphs and its applications in graph drawing problems**, G. Kant, X. He, http://www.sciencedirect.com/science/article/pii/S030439759500257X

- Process nodes of degree < 4
  → Integrate these nodes into adjacent one
- Degree 3:



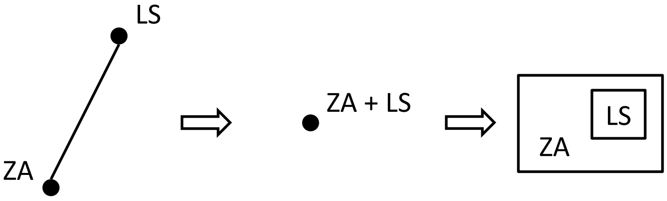
→ BE becomes L-shaped in final cartogram

- Degree 2:



→ IN becomes C-shaped in final cartogram

- Degree 1:

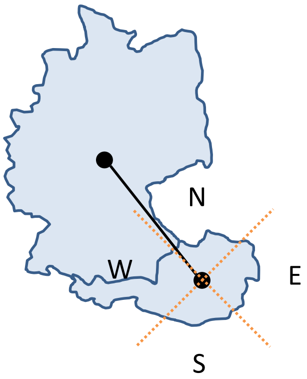


→ ZA becomes O-shaped in final cartogram

- Disconnected regions
  → Choose one region to represent all

2. Directed regular edge labels
   The directed edge labels give an adjacency direction between neighboring vertices of F. That follow from the relative geographic position of the respective regions.

Heuristic
Consider the line through the centers of mass of the regions and its compass direction



We orient the directions from
south → north (T_1) from west → east (T_2)

OBSERVATION

A face graph F with directed regular edge labeling can be represented by a rectangular dual iff:

1. Every internal region has at least 1 neighbor on each compass direction
2. When traversing the regions of any node v in clockwise order, starting at the west most northern neighbor, we encounter:
    - All north neighbors
    - All east neighbors
    - All south neighbors
    - All west neighbors

    → "Realizable" edge labeling

3. Rectangular layout
    - We add 4 outer nodes to F: N, E, W, S and edges: N-E, N-W, S-E, S-W
    - If necessary add "sea" regions to preserve shape and adjacencies
      → New graph G
    - Construct x-coordinates
        a) Let G_1 be the graph consisting of all vertices of G and all edges of T_1 plus the 4 exterior edges with directions $S \rightarrow W,\ W \rightarrow N,\ S \rightarrow E,\ E \rightarrow N$
        $\Rightarrow G_1$ is a planar st-graph with source S and sink N
        b) Associated dual st-graph G_1^*: Let the source of G_1^* be the left face of W^* and the sink the right face of E^*. For each node $f \in G_1^*$ let $d_1(f)$ be the length of the longest path from W^* to f; Let $D_1 \coloneqq d_1(E^*)$.
        For each interior vertex v:
        $$x_{LEFT}(v) = d_1(LEFT(v))$$
        $$x_{RIGHT}(v) = d_1(RIGHT(v))$$
        For the 4 exterior vertices:
        $$x_{LEFT}(W) = 0$$
        $$x_{RIGHT}(W) = 1$$
        $$x_{LEFT}(E) = D_1 - 1$$
        $$x_{RIGHT}(E) = D_1$$
        $$x_{LEFT}(S) = 1$$
        $$x_{RIGHT}(S) = D_1 - 1$$
        $$X_{LEFT}(N) = 1$$
        $$x_{RIGHT}(N) = D_1 - 1$$

- <u>Constructing the y-coordinates</u>
  Similar with G_2. G_2 consists of all vertices of G, all edges of T_2,plus the exterior edges $E \rightarrow S, S \rightarrow E, W \rightarrow N, N \rightarrow E$
  G_2: source = W, sink = E

THEOREM
Let G be a graph with realizable directed edge labeling we assign to each vertex $v \in G$ the rectangle $f(v)$ bounded by lines:

$$X_{LEFT}(v), \qquad x_{RIGHT}(v)$$
$$y_{LOW}(v), \qquad y_{HIGH}(v)$$

Then the set $\{f(v)|v \in G\}$ forms a rectangular dual of G. If a relizable directed edge labeling is given, a rectangular dual takes $O(n)$ time.

6.3 Main cartogram algorithm

See handout!