Vorlesung Graphenzeichnen

Order in the Underground *or* How to automate drawing metro maps?

Martin Nöllenburg

Institute of Theoretical Informatics

Karlsruhe Institute of Technology

Karlsruhe Institute of Technology

07.12.2011

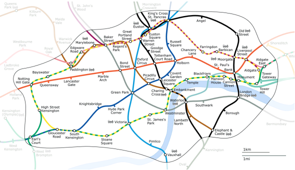
Outline

- Modeling the Metro Map Problem
  - What is a Metro Map?
  - Hard and Soft Constraints
- NP-Hardness: Bad News—Nice Proof
  - Rectilinear vs. Octilinear Drawing
  - Reduction from PLANAR 3-SAT
- MIP Formulation & Experiments
  - Mixed-Integer Programming Formulation
  - Labeling
  - Experiments

Outline

- Modeling the Metro Map Problem
  - What is a Metro Map?
  - Hard and Soft Constraints
- NP-Hardness: Bad News—Nice Proof
  - Rectilinear vs. Octilinear Drawing
  - Reduction from PLANAR 3-SAT
- MIP Formulation & Experiments
  - Mixed-Integer Programming Formulation
  - Labeling
  - Experiments

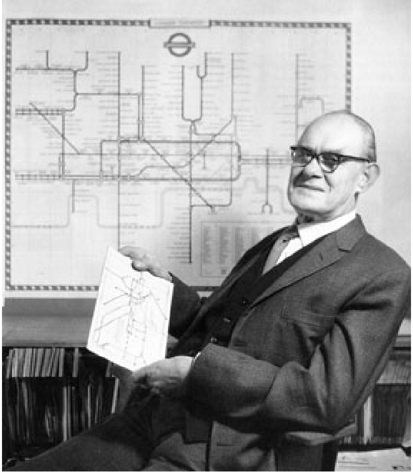
What is a Metro Map?



- schematic diagram for public transport
- visualizes lines and stations
- goal: ease navigation for passengers
  - “How do I get from A to B?”
  - “Where to get off and change trains?”
- distorts geometry and scale
- improves readability
- compromise between
  schematic road map $\leftrightarrow$ abstract graph

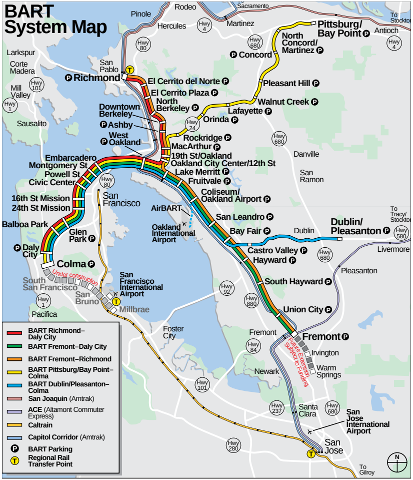
Why Automate Drawing Metro Maps?

- current maps designed manually



Why Automate Drawing Metro Maps?

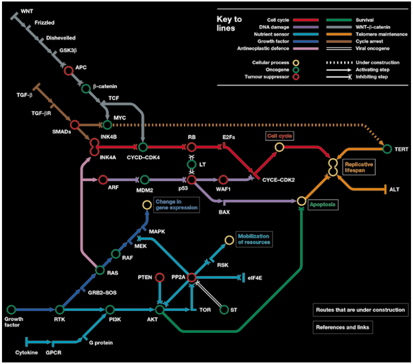
- current maps designed manually
- assist graphic designers to improve/extend maps



Why Automate Drawing Metro Maps?

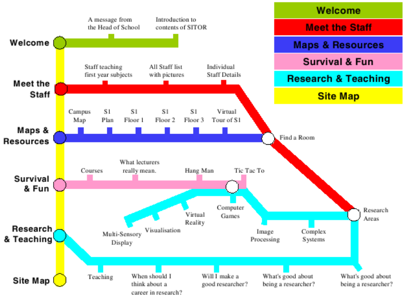
- current maps designed manually
- assist graphic designers to improve/extend maps
- metro map metaphor
  - metabolic pathways

[Hahn, Weinberg '02]



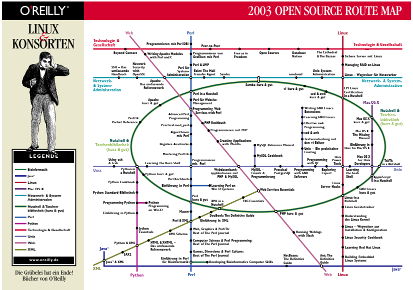
Why Automate Drawing Metro Maps?

- current maps designed manually
- assist graphic designers to improve/extend maps
- metro map metaphor
  - metabolic pathways [Hahn, Weinberg '02]
  - web page maps [Nesbitt '04]



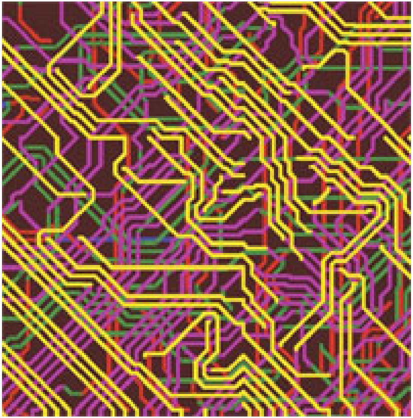
Why Automate Drawing Metro Maps?

- current maps designed manually
- assist graphic designers to improve/extend maps
- metro map metaphor
    - metabolic pathways [Hahn, Weinberg '02]
    - web page maps [Nesbitt '04]
    - product lines [O'Reilly '03]



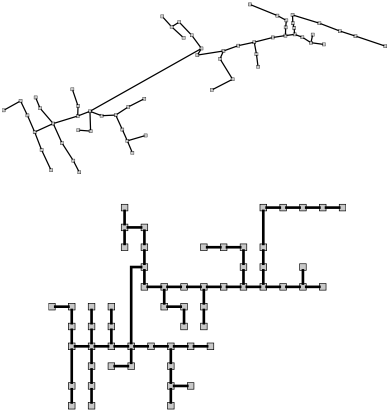
Why Automate Drawing Metro Maps?

- current maps designed manually
- assist graphic designers to improve/extend maps
- metro map metaphor
  - metabolic pathways [Hahn, Weinberg '02]
  - web page maps [Nesbitt '04]
  - product lines [O'Reilly '03]
- VLSI: X-architecture



Why Automate Drawing Metro Maps?

- current maps designed manually
- assist graphic designers to improve/extend maps
- metro map metaphor
  - metabolic pathways [Hahn, Weinberg '02]
  - web page maps [Nesbitt '04]
  - product lines [O'Reilly '03]
- VLSI: X-architecture
- redrawing sketches [Brandes et al. '03]



More Formally

The Metro Map Problem

Given: planar embedded graph $G = (V, E)$, $V \subset \mathbb{R}^2$,
line cover $\mathcal{L}$ of paths or cycles in G (the metro lines),

Goal: draw G and $\mathcal{L}$ nicely.

More Formally

The Metro Map Problem

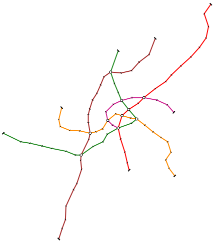
Given: planar embedded graph $G = (V, E)$, $V \subset \mathbb{R}^2$,
line cover $\mathcal{L}$ of paths or cycles in G (the metro lines),

Goal: draw G and $\mathcal{L}$ nicely.

- What is a nice drawing?
- Look at real-world metro maps drawn by graphic designers and model their design principles as
  - *hard* constraints – must be fulfilled,
  - *soft* constraints – should hold as tightly as possible.

Hard Constraints

(H1) preserve embedding of G



Hard Constraints

(H1) preserve embedding of G

(H2) draw all edges as octilinear line segments, i.e. horizontal, vertical or diagonal (45 degrees)

Hard Constraints

(H1) preserve embedding of G

(H2) draw all edges as octilinear line segments,
i.e. horizontal, vertical or diagonal (45 degrees)

(H3) draw each edge e with length $\geq \ell_e$

Hard Constraints

(H1) preserve embedding of G

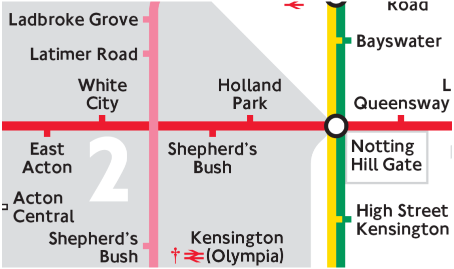
(H2) draw all edges as octilinear line segments, i.e. horizontal, vertical or diagonal (45 degrees)

(H3) draw each edge e with length $\geq \ell_e$

(H4) keep edges d_{min} away from non-incident edges ($\rightarrow$ planar)

Soft Constraints

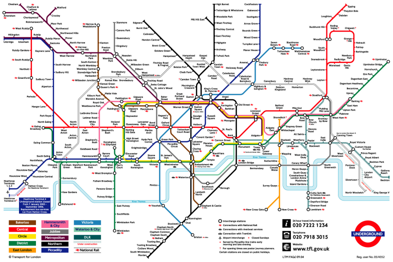
(S1) draw metro lines with few bends



Soft Constraints

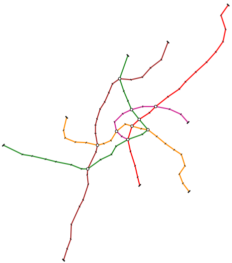
(S1) draw metro lines with few bends

(S2) keep total edge length small



Soft Constraints

- (S1) draw metro lines with few bends
- (S2) keep total edge length small
- (S3) draw each octilinear edge similar to its geographical orientation: keep relative position of adjacent vertices



Outline

- Modeling the Metro Map Problem
  - What is a Metro Map?
  - Hard and Soft Constraints
- NP-Hardness: Bad News—Nice Proof
  - Rectilinear vs. Octilinear Drawing
  - Reduction from PLANAR 3-SAT
- MIP Formulation & Experiments
  - Mixed-Integer Programming Formulation
  - Labeling
  - Experiments

A Related Problem

RECTILINEARGRAPHDRAWING Decision Problem

Given a planar embedded graph G with max degree 4.
Is there a drawing of G that

- preserves the embedding,
- uses straight-line edges,
- is rectilinear?

A Related Problem

RECTILINEARGRAPHDRAWING Decision Problem

Given a planar embedded graph G with max degree 4.
Is there a drawing of G that

- preserves the embedding,
- uses straight-line edges,
- is rectilinear?

Theorem (Tamassia SIAMJComp'87)

RECTILINEARGRAPHDRAWING *can be solved efficiently.*

A Related Problem

RECTILINEARGRAPHDRAWING Decision Problem

Given a planar embedded graph G with max degree 4.
Is there a drawing of G that

- preserves the embedding,
- uses straight-line edges,
- is rectilinear?

Theorem (Tamassia SIAMJComp'87)

RECTILINEARGRAPHDRAWING *can be solved efficiently.*

Proof.

By reduction to a flow problem. □

A Related Problem

RECTILINEARGRAPHDRAWING Decision Problem

Given a planar embedded graph G with max degree 4.
Is there a drawing of G that

- preserves the embedding,
- uses straight-line edges,
- is rectilinear?

Theorem (Tamassia SIAMJComp'87)

RECTILINEARGRAPHDRAWING *can be solved efficiently.*

Proof.

By reduction to a flow problem. □

Our Problem

METROMAPLAYOUT Decision Problem

Given a planar embedded graph G with max degree 8.
Is there a drawing of G that

- preserves the embedding,
- uses straight-line edges,
- is octilinear?

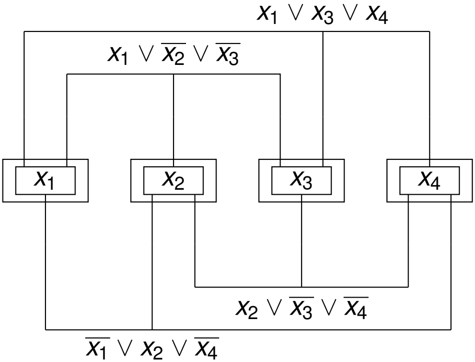
Theorem

METROMAPLAYOUT *is NP-hard.*

Proof.

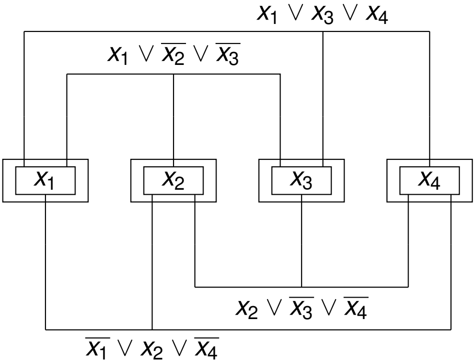
By Reduction from PLANAR 3-SAT to METROMAPLAYOUT. □

Outline of the Reduction



Input: planar 3-SAT formula $\varphi =$
$(x_1 \vee x_3 \vee x_4) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge \dots$

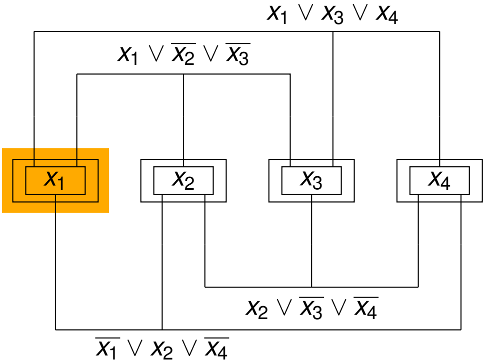
Outline of the Reduction



Input: planar 3-SAT formula $\varphi = (x_1 \vee x_3 \vee x_4) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge \dots$

Goal: planar embedded graph G_φ with:
G_φ has a metro map drawing $\Leftrightarrow$ φ satisfiable.

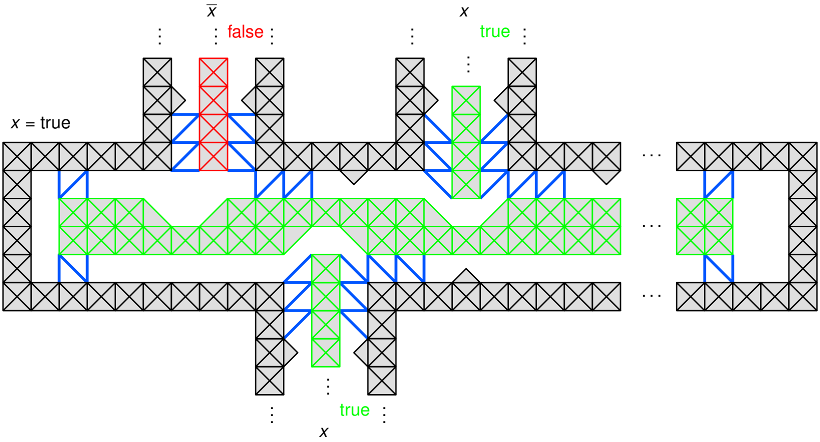
Outline of the Reduction



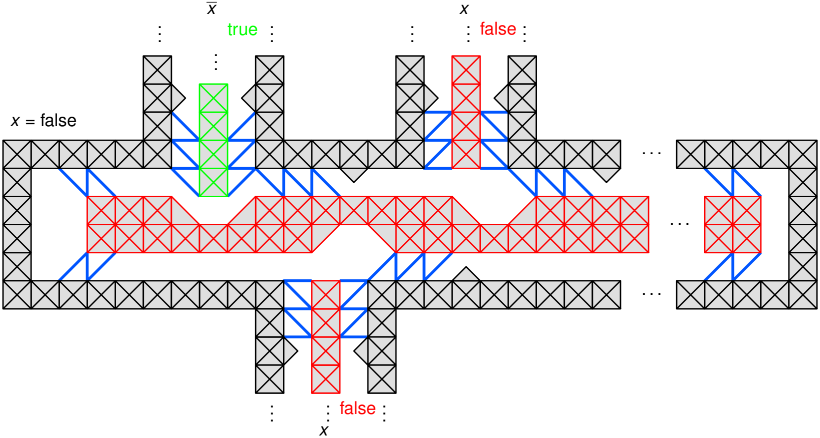
Input: planar 3-SAT formula $\varphi = (x_1 \lor x_3 \lor x_4) \land (x_1 \lor \overline{x_2} \lor \overline{x_3}) \land \ldots$

Goal: planar embedded graph G_φ with:
G_φ has a metro map drawing $\Leftrightarrow$ φ satisfiable.

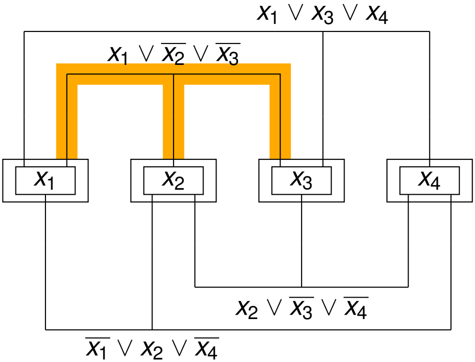
Variable Gadget



Variable Gadget



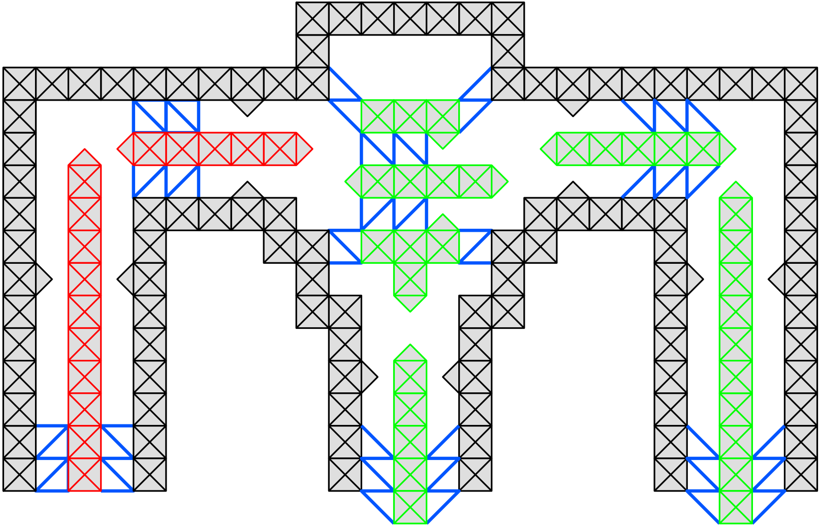
Outline of the Reduction



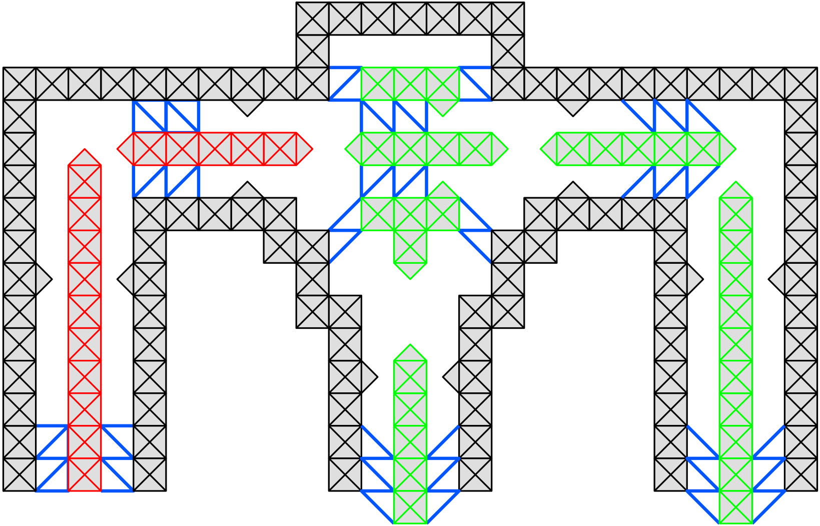
Input: planar 3-SAT formula $\varphi = (x_1 \vee x_3 \vee x_4) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge \dots$

Goal: planar embedded graph G_φ with:
G_φ has a metro map drawing $\Leftrightarrow$ φ satisfiable.

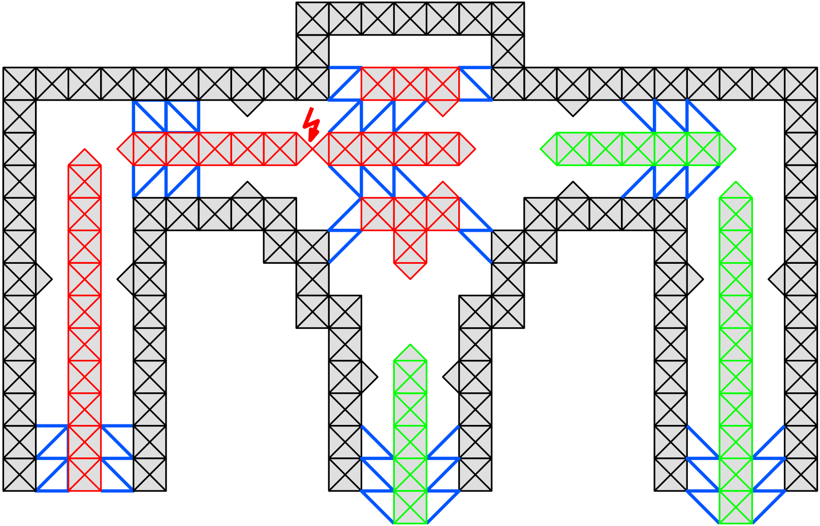
Clause Gadget



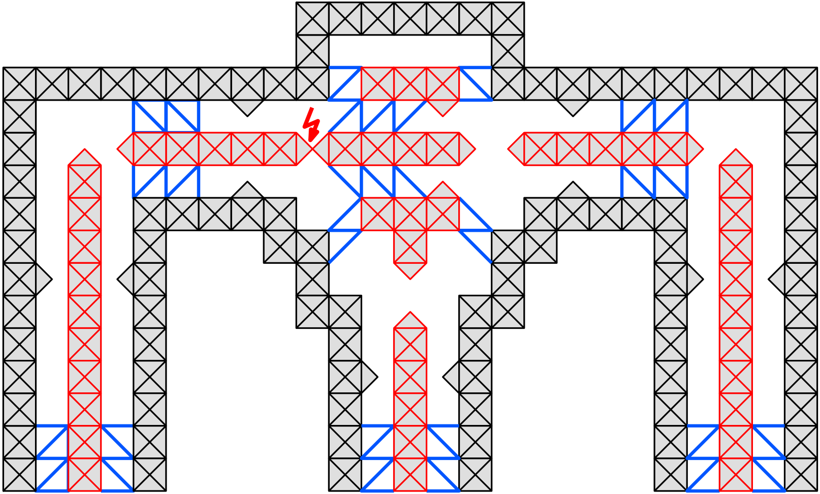
Clause Gadget



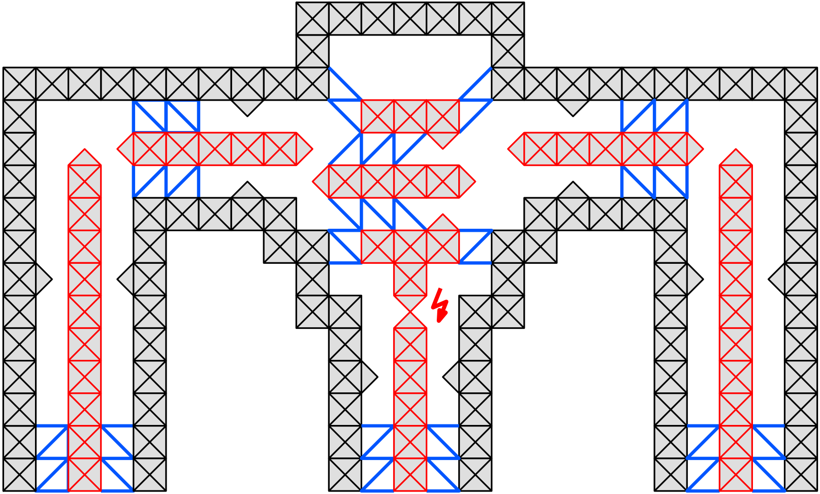
Clause Gadget



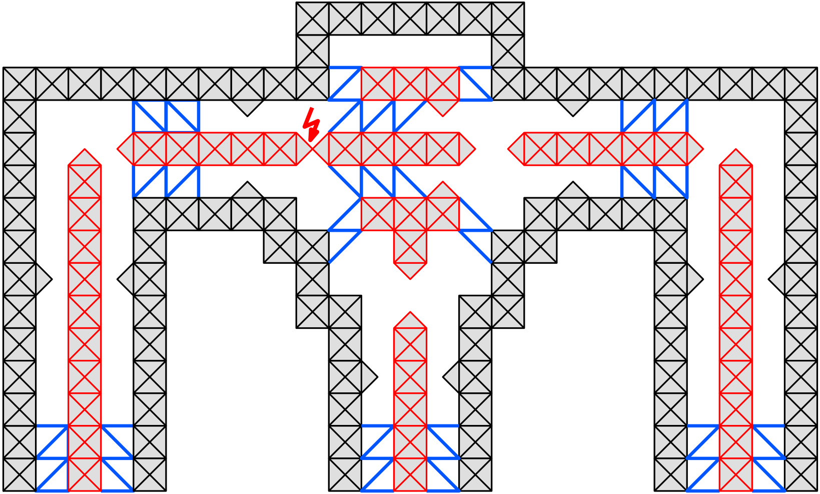
Clause Gadget



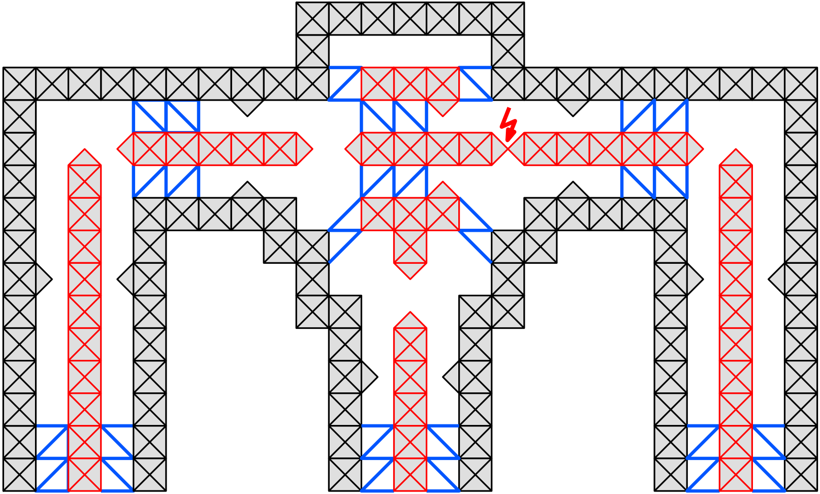
Clause Gadget



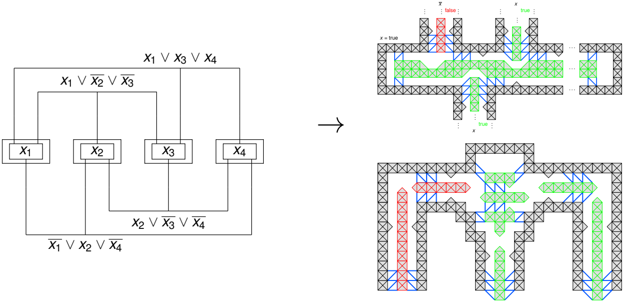
Clause Gadget



Clause Gadget



Summary of the Reduction



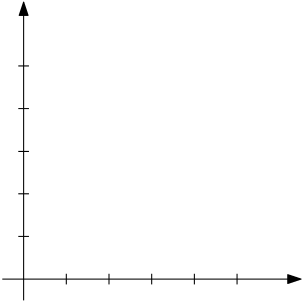
- Indeed we have:
  - φ satisfiable $\Rightarrow$ corresponding MM drawing of G_φ
  - G_φ has MM drawing $\Rightarrow$ satisfying truth assignment of φ

Outline

- Modeling the Metro Map Problem
  - What is a Metro Map?
  - Hard and Soft Constraints
- NP-Hardness: Bad News—Nice Proof
  - Rectilinear vs. Octilinear Drawing
  - Reduction from PLANAR 3-SAT
- MIP Formulation & Experiments
  - Mixed-Integer Programming Formulation
  - Labeling
  - Experiments

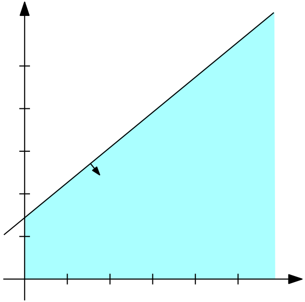
Mathematical Programming

- Linear Programming: efficient optimization method for
  - linear constraints
  - linear objective function
  - real-valued variables



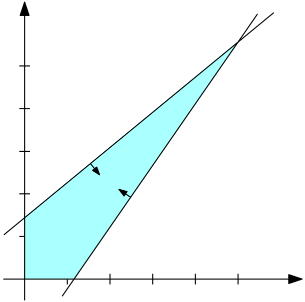
Mathematical Programming

- Linear Programming: efficient optimization method for
  - linear constraints
  - linear objective function
  - real-valued variables
  - example:
    maximize $x + 2y$
    subject to
    $y \leq 0.9x + 1.5$
    $y \geq 1.4x - 1.3$



Mathematical Programming

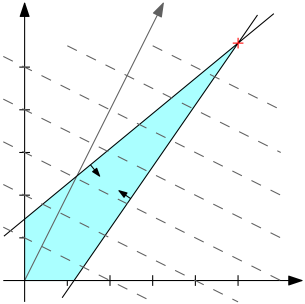
- Linear Programming: efficient optimization method for
  - linear constraints
  - linear objective function
  - real-valued variables
  - example:
    maximize $x + 2y$
    subject to
    $y \leq 0.9x + 1.5$
    $y \geq 1.4x - 1.3$



Mathematical Programming

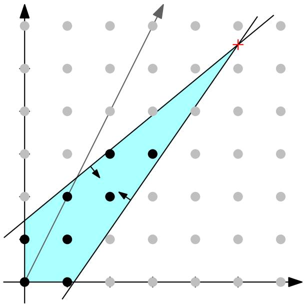
- Linear Programming: efficient optimization method for
  - linear constraints
  - linear objective function
  - real-valued variables
  - example:

    maximize $x + 2y$
    subject to
    $y \leq 0.9x + 1.5$
    $y \geq 1.4x - 1.3$



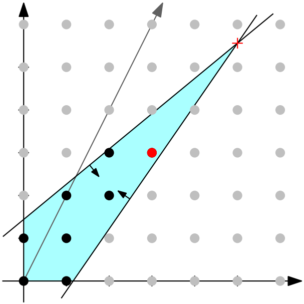
Mathematical Programming

- Linear Programming: efficient optimization method for
  - linear constraints
  - linear objective function
  - real-valued variables
- Mixed-Integer Programming (MIP)
  - allows also integer variables
  - NP-hard in general
  - still practical method for many hard optimization problems



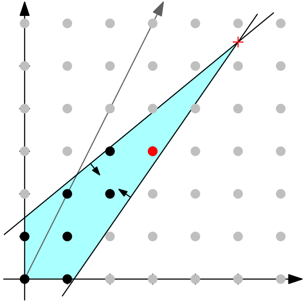
Mathematical Programming

- Linear Programming: efficient optimization method for
  - linear constraints
  - linear objective function
  - real-valued variables
- Mixed-Integer Programming (MIP)
  - allows also integer variables
  - NP-hard in general
  - still practical method for many hard optimization problems



Mathematical Programming

- Linear Programming: efficient optimization method for
  - linear constraints
  - linear objective function
  - real-valued variables
- Mixed-Integer Programming (MIP)
  - allows also integer variables
  - NP-hard in general
  - still practical method for many hard optimization problems



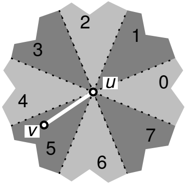
Theorem (GD'05 / TVCG'11)

The metro map layout problem can be formulated as a MIP s.th.

hard constraints $\rightarrow$ *linear constraints*

soft constraints $\rightarrow$ *objective function*

Definitions: Sectors and Coordinates

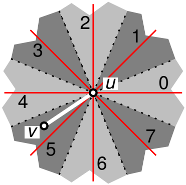


Sectors

– for each vtx. u partition plane into sectors 0–7

- here: $\mathrm{sec}(u, v) = 5$ (input)

Definitions: Sectors and Coordinates



Sectors

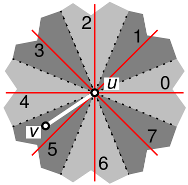
– for each vtx. u partition plane into sectors 0–7

- here: $\mathrm{sec}(u, v) = 5$ (input)

– number octilinear edge directions accordingly

- e.g. $\mathrm{dir}(u, v) = 4$ (output)

Definitions: Sectors and Coordinates



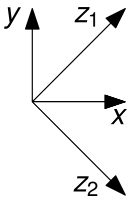
Sectors

– for each vtx. u partition plane into sectors 0–7

- here: $\text{sec}(u, v) = 5$ (input)

– number octilinear edge directions accordingly

- e.g. $\text{dir}(u, v) = 4$ (output)



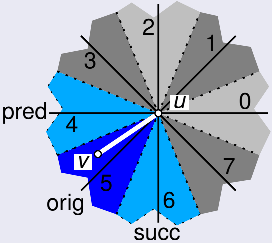
Coordinates

assign z_1- and z_2-coordinates to each vertex v:

- $z_1(v) = x(v) + y(v)$
- $z_2(v) = x(v) - y(v)$

Octilinearity and Relative Position (H2, H3, S3)

Goal

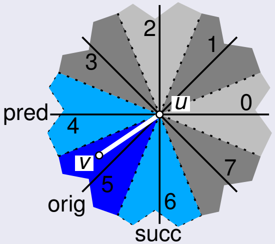


Draw edge *uv*

- octilinearly
- with minimum length ℓ_{uv}
- restricted to 3 directions

Octilinearity and Relative Position (H2, H3, S3)

Goal



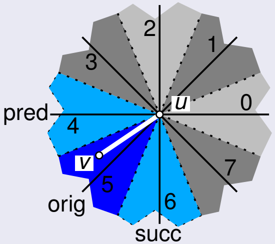
Draw edge *uv*

- octilinearly
- with minimum length ℓ_{uv}
- restricted to 3 directions

How to model this using linear constraints?

Octilinearity and Relative Position (H2, H3, S3)

Goal



Draw edge *uv*

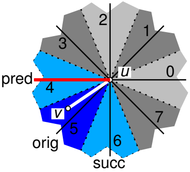
- octilinearly
- with minimum length ℓ_{uv}
- restricted to 3 directions

How to model this using linear constraints?

Binary Variables

$$\alpha_{\text{pred}}(u,v) + \alpha_{\text{orig}}(u,v) + \alpha_{\text{succ}}(u,v) = 1$$

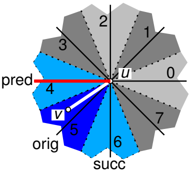
Octilinearity and Relative Position



Predecessor Sector

$$
\begin{aligned}
y(u) - y(v) &\leq M(1 - \alpha_{\text{pred}}(u, v)) \\
-y(u) + y(v) &\leq M(1 - \alpha_{\text{pred}}(u, v)) \\
x(u) - x(v) &\geq -M(1 - \alpha_{\text{pred}}(u, v)) + \ell_{uv}
\end{aligned}
$$

Octilinearity and Relative Position

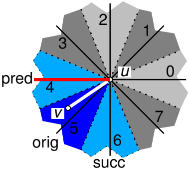


Predecessor Sector

$$
\begin{aligned}
y(u) - y(v) &\leq M(1 - \alpha_{\text{pred}}(u,v)) \\
-y(u) + y(v) &\leq M(1 - \alpha_{\text{pred}}(u,v)) \\
x(u) - x(v) &\geq -M(1 - \alpha_{\text{pred}}(u,v)) + \ell_{uv}
\end{aligned}
$$

How does this work?

Octilinearity and Relative Position



Predecessor Sector

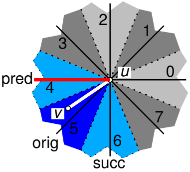
$$\begin{aligned} y(u) - y(v) &\leq M(1 - \alpha_{\text{pred}}(u, v)) \\ -y(u) + y(v) &\leq M(1 - \alpha_{\text{pred}}(u, v)) \\ x(u) - x(v) &\geq -M(1 - \alpha_{\text{pred}}(u, v)) + \ell_{uv} \end{aligned}$$

How does this work?

Case 1: $\alpha_{\text{pred}}(u, v) = 0$

$$\begin{aligned} y(u) - y(v) &\leq M \\ -y(u) + y(v) &\leq M \\ x(u) - x(v) &\geq \ell_{uv} - M \end{aligned}$$

Octilinearity and Relative Position



Predecessor Sector

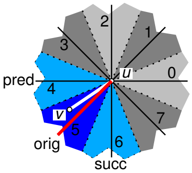
$$\begin{aligned} y(u) - y(v) &\leq M(1 - \alpha_{\text{pred}}(u,v)) \\ -y(u) + y(v) &\leq M(1 - \alpha_{\text{pred}}(u,v)) \\ x(u) - x(v) &\geq -M(1 - \alpha_{\text{pred}}(u,v)) + \ell_{uv} \end{aligned}$$

How does this work?

Case 2: $\alpha_{\text{pred}}(u,v) = 1$

$$\begin{aligned} y(u) - y(v) &\leq 0 \\ -y(u) + y(v) &\leq 0 \\ x(u) - x(v) &\geq \ell_{uv} \end{aligned}$$

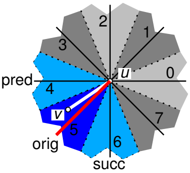
Octilinearity and Relative Position (H2, H3, S3)



Original Sector

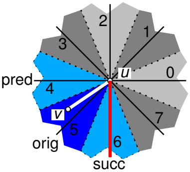
$$\begin{aligned} z_2(u) - z_2(v) &\leq M(1 - \alpha_{\text{orig}}(u,v)) \\ -z_2(u) + z_2(v) &\leq M(1 - \alpha_{\text{orig}}(u,v)) \\ z_1(u) - z_1(v) &\geq -M(1 - \alpha_{\text{orig}}(u,v)) + 2\ell_{uv} \end{aligned}$$

Octilinearity and Relative Position (H2, H3, S3)



Original Sector

$$
\begin{aligned}
z_2(u) - z_2(v) &\leq M(1 - \alpha_{orig}(u,v)) \\
-z_2(u) + z_2(v) &\leq M(1 - \alpha_{orig}(u,v)) \\
z_1(u) - z_1(v) &\geq -M(1 - \alpha_{orig}(u,v)) + 2\ell_{uv}
\end{aligned}
$$



Successor Sector

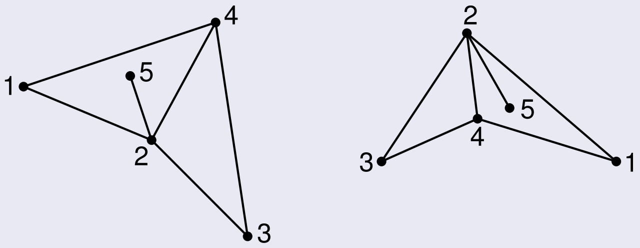
$$
\begin{aligned}
x(u) - x(v) &\leq M(1 - \alpha_{succ}(u,v)) \\
-x(u) + x(v) &\leq M(1 - \alpha_{succ}(u,v)) \\
y(u) - y(v) &\geq -M(1 - \alpha_{succ}(u,v)) + \ell_{uv}
\end{aligned}
$$

Preserving the Embedding (H1)

Definition

Two planar drawings of G have the same *embedding* if the induced orderings on the neighbors of each vertex are equal.

Same Embedding

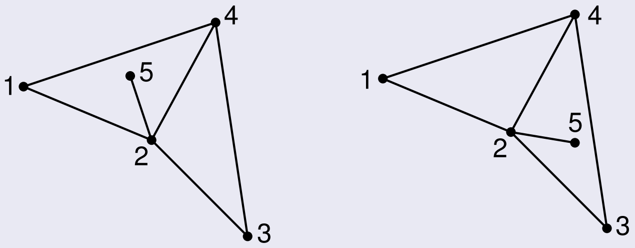


Preserving the Embedding (H1)

Definition

Two planar drawings of G have the same *embedding* if the induced orderings on the neighbors of each vertex are equal.

Different Embeddings



Preserving the Embedding (H1)

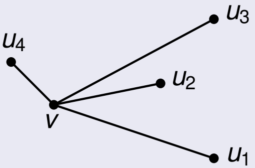
Constraints (Example)

- $N(v) = \{u_1, u_2, u_3, u_4\}$
- circular input order: $u_1 < u_2 < u_3 < u_4 < u_1$

All but one of the following inequalities must hold

$$\mathrm{dir}(v, u_1) < \mathrm{dir}(v, u_2) < \mathrm{dir}(v, u_3) < \mathrm{dir}(v, u_4) < \mathrm{dir}(v, u_1)$$

Input



Preserving the Embedding (H1)

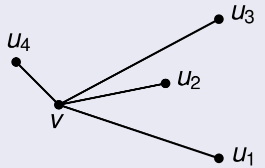
Constraints (Example)

- $N(v) = \{u_1, u_2, u_3, u_4\}$
- circular input order: $u_1 < u_2 < u_3 < u_4 < u_1$

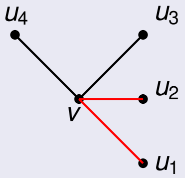
All but one of the following inequalities must hold

$$\text{dir}(v, u_1) \not< \text{dir}(v, u_2) < \text{dir}(v, u_3) < \text{dir}(v, u_4) < \text{dir}(v, u_1)$$

Input



Output

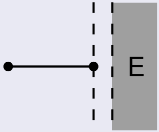


Planarity (H4)

Observation

For octilinear, straight edge e_1 non-incident edge e_2 must be placed $d_{\min}$ to the

- east, northeast, north, northwest, west, southwest, south, or southeast

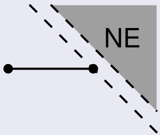


Planarity (H4)

Observation

For octilinear, straight edge e_1 non-incident edge e_2 must be placed $d_{\min}$ to the

- east, northeast, north, northwest, west, southwest, south, or southeast

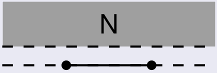


Planarity (H4)

Observation

For octilinear, straight edge e_1 non-incident edge e_2 must be placed $d_{\min}$ to the

- east, northeast, north, northwest, west, southwest, south, or southeast

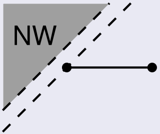


Planarity (H4)

Observation

For octilinear, straight edge e_1 non-incident edge e_2 must be placed $d_{\min}$ to the

- east, northeast, north, northwest, west, southwest, south, or southeast

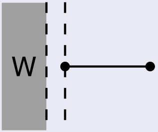


Planarity (H4)

Observation

For octilinear, straight edge e_1 non-incident edge e_2 must be placed d_{min} to the

- east, northeast, north, northwest, west, southwest, south, or southeast

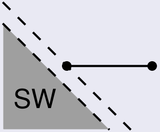


Planarity (H4)

Observation

For octilinear, straight edge e_1 non-incident edge e_2 must be placed d_{min} to the

- east, northeast, north, northwest, west, southwest, south, or southeast

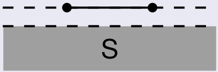


Planarity (H4)

Observation

For octilinear, straight edge e_1 non-incident edge e_2 must be placed $d_{\min}$ to the

- east, northeast, north, northwest, west, southwest, south, or southeast

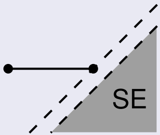


Planarity (H4)

Observation

For octilinear, straight edge e_1 non-incident edge e_2 must be placed d_{min} to the

- east, northeast, north, northwest, west, southwest, south, or southeast

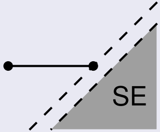


Planarity (H4)

Observation

For octilinear, straight edge e_1 non-incident edge e_2 must be placed $d_{\min}$ to the

- east, northeast, north, northwest, west, southwest, south, or southeast



Constraints

- model as MIP with binary variables
  $\alpha_E + \alpha_{NE} + \alpha_N + \alpha_{NW} + \alpha_W + \alpha_{SW} + \alpha_S + \alpha_{SE} \geq 1$
- required for each pair of non-incident edges

Objective Function

Objective Function

- corresponds to soft constraints (S1)–(S3)
- weighted sum of individual cost functions

minimize $\lambda_{\text{bends}}\, \text{cost}_{\text{bends}} + \lambda_{\text{length}}\, \text{cost}_{\text{length}} + \lambda_{\text{relpos}}\, \text{cost}_{\text{relpos}}$

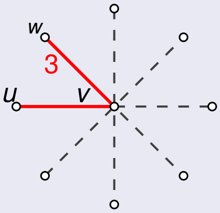
Objective Function

Objective Function

- corresponds to soft constraints (S1)–(S3)
- weighted sum of individual cost functions

minimize $\lambda_{\text{bends}}\,\text{cost}_{\text{bends}} + \lambda_{\text{length}}\,\text{cost}_{\text{length}} + \lambda_{\text{relpos}}\,\text{cost}_{\text{relpos}}$

Line Bends (S1)



Edges uv and vw on a metro line $L \in \mathcal{L}$

- draw as straight as possible
- increase cost bend(u, v, w) for increasing acuteness of $\angle(\overline{uv}, \overline{vw})$

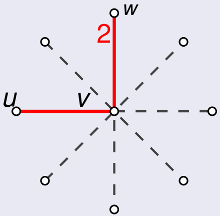
Objective Function

Objective Function

- corresponds to soft constraints (S1)–(S3)
- weighted sum of individual cost functions

minimize $\lambda_{\text{bends}}\,\text{cost}_{\text{bends}} + \lambda_{\text{length}}\,\text{cost}_{\text{length}} + \lambda_{\text{relpos}}\,\text{cost}_{\text{relpos}}$

Line Bends (S1)



Edges uv and vw on a metro line $L \in \mathcal{L}$

- draw as straight as possible
- increase cost bend(u, v, w) for increasing acuteness of $\angle(\overline{uv}, \overline{vw})$

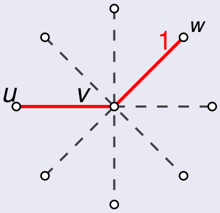
Objective Function

Objective Function

- corresponds to soft constraints (S1)–(S3)
- weighted sum of individual cost functions

minimize $\lambda_{\text{bends}}\,\text{cost}_{\text{bends}} + \lambda_{\text{length}}\,\text{cost}_{\text{length}} + \lambda_{\text{relpos}}\,\text{cost}_{\text{relpos}}$

Line Bends (S1)



Edges uv and vw on a metro line $L \in \mathcal{L}$

- draw as straight as possible
- increase cost bend(u, v, w) for increasing acuteness of $\angle(\overline{uv}, \overline{vw})$

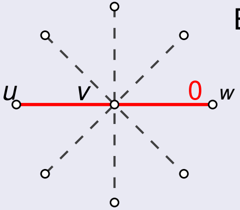
Objective Function

Objective Function

- corresponds to soft constraints (S1)–(S3)
- weighted sum of individual cost functions

minimize $\lambda_{\text{bends}}\,\text{cost}_{\text{bends}} + \lambda_{\text{length}}\,\text{cost}_{\text{length}} + \lambda_{\text{relpos}}\,\text{cost}_{\text{relpos}}$

Line Bends (S1)



Edges uv and vw on a metro line $L \in \mathcal{L}$

- draw as straight as possible
- increase cost bend(u, v, w) for increasing acuteness of $\angle(\overline{uv}, \overline{vw})$

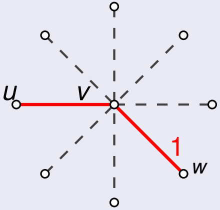
Objective Function

Objective Function

- corresponds to soft constraints (S1)–(S3)
- weighted sum of individual cost functions

minimize $\lambda_{\text{bends}}\ \text{cost}_{\text{bends}} + \lambda_{\text{length}}\ \text{cost}_{\text{length}} + \lambda_{\text{relpos}}\ \text{cost}_{\text{relpos}}$

Line Bends (S1)



Edges *uv* and *vw* on a metro line $L \in \mathcal{L}$

- draw as straight as possible
- increase cost bend(u, v, w) for increasing acuteness of $\angle(\overline{uv}, \overline{vw})$

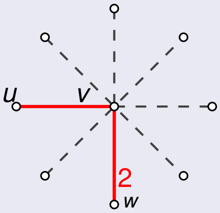
Objective Function

Objective Function

- corresponds to soft constraints (S1)–(S3)
- weighted sum of individual cost functions

minimize $\lambda_{\text{bends}}\,\text{cost}_{\text{bends}} + \lambda_{\text{length}}\,\text{cost}_{\text{length}} + \lambda_{\text{relpos}}\,\text{cost}_{\text{relpos}}$

Line Bends (S1)



Edges uv and vw on a metro line $L \in \mathcal{L}$

- draw as straight as possible
- increase cost bend(u, v, w) for increasing acuteness of $\angle(\overline{uv}, \overline{vw})$

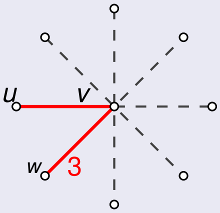
Objective Function

Objective Function

- corresponds to soft constraints (S1)–(S3)
- weighted sum of individual cost functions

minimize $\lambda_{\text{bends}}\,\text{cost}_{\text{bends}} + \lambda_{\text{length}}\,\text{cost}_{\text{length}} + \lambda_{\text{relpos}}\,\text{cost}_{\text{relpos}}$

Line Bends (S1)



Edges uv and vw on a metro line $L \in \mathcal{L}$

- draw as straight as possible
- increase cost bend(u, v, w) for increasing acuteness of $\angle(\overline{uv}, \overline{vw})$

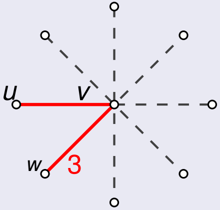
Objective Function

Objective Function

- corresponds to soft constraints (S1)–(S3)
- weighted sum of individual cost functions

minimize $\lambda_{\text{bends}}\, \text{cost}_{\text{bends}} + \lambda_{\text{length}}\, \text{cost}_{\text{length}} + \lambda_{\text{relpos}}\, \text{cost}_{\text{relpos}}$

Line Bends (S1)



Edges uv and vw on a metro line $L \in \mathcal{L}$

- draw as straight as possible
- increase cost bend(u, v, w) for increasing acuteness of $\angle(\overline{uv}, \overline{vw})$

$$\text{cost}_{\text{bends}} = \sum_{L \in \mathcal{L}} \sum_{uv, vw \in L} \text{bend}(u, v, w)$$

Objective Function

Total Edge Length (S2)

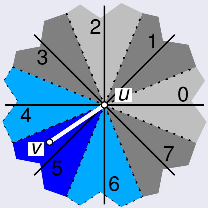
$$\text{cost}_{\text{length}} = \sum_{uv \in E} \text{length}(\overline{uv})$$

Objective Function

Total Edge Length (S2)

$$\text{cost}_{\text{length}} = \sum_{uv \in E} \text{length}(\overline{uv})$$

Relative Position (S3)



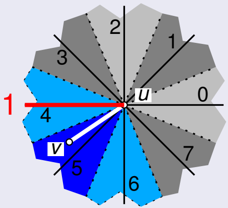
- only three directions possible

Objective Function

Total Edge Length (S2)

$$\mathrm{cost}_{\mathrm{length}} = \sum_{uv \in E} \mathrm{length}(\overline{uv})$$

Relative Position (S3)



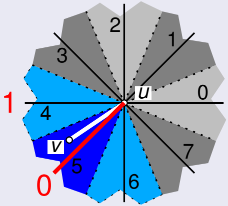
- only three directions possible
- charge 1 if edge deviates from original sector

Objective Function

Total Edge Length (S2)

$$\text{cost}_{\text{length}} = \sum_{uv \in E} \text{length}(\overline{uv})$$

Relative Position (S3)



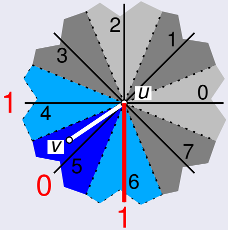
- only three directions possible
- charge 1 if edge deviates from original sector

Objective Function

Total Edge Length (S2)

$$\text{cost}_{\text{length}} = \sum_{uv \in E} \text{length}(\overline{uv})$$

Relative Position (S3)



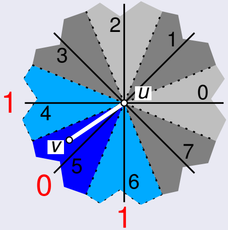
- only three directions possible
- charge 1 if edge deviates from original sector

Objective Function

Total Edge Length (S2)

$$\text{cost}_{\text{length}} = \sum_{uv \in E} \text{length}(\overline{uv})$$

Relative Position (S3)



- only three directions possible
- charge 1 if edge deviates from original sector

$$\text{cost}_{\text{relpos}} = \sum_{uv \in E} \text{relpos}(uv)$$

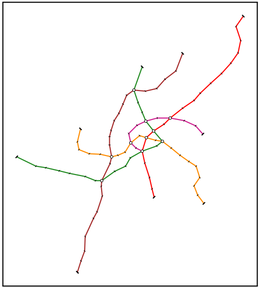
Effects of the Soft Constraints

Objective Function

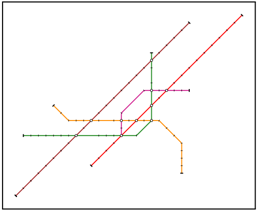
- corresponds to soft constraints (S1)–(S3)
- weighted sum of individual cost functions

minimize $\lambda_{\text{bends}}\,\text{cost}_{\text{bends}} + \lambda_{\text{length}}\,\text{cost}_{\text{length}} + \lambda_{\text{relpos}}\,\text{cost}_{\text{relpos}}$

Input



Output



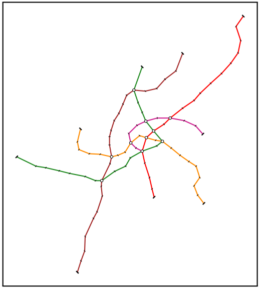
Effects of the Soft Constraints

Objective Function

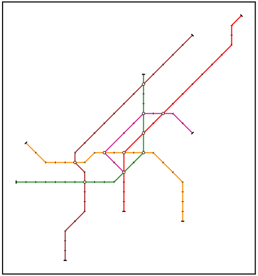
- corresponds to soft constraints (S1)–(S3)
- weighted sum of individual cost functions

minimize $\lambda_{\text{bends}} \operatorname{cost}_{\text{bends}} + \lambda_{\text{length}} \operatorname{cost}_{\text{length}} + \lambda_{\text{relpos}} \operatorname{cost}_{\text{relpos}}$

Input



Output



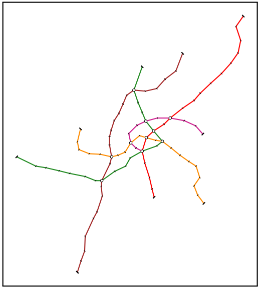
Effects of the Soft Constraints

Objective Function

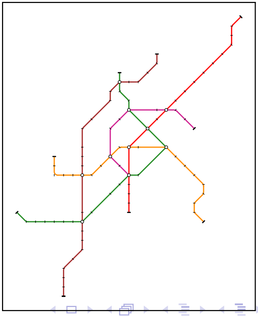
- corresponds to soft constraints (S1)–(S3)
- weighted sum of individual cost functions

minimize $\lambda_{\text{bends}}\,\text{cost}_{\text{bends}} + \lambda_{\text{length}}\,\text{cost}_{\text{length}} + \lambda_{\text{relpos}}\,\text{cost}_{\text{relpos}}$

Input



Output



Summary of the MIP

- hard constraints:
  - octilinearity
  - minimum edge length
  - (partially) relative position
  - preservation of embedding
  - planarity

Summary of the MIP

- hard constraints:
  - octilinearity
  - minimum edge length
  - (partially) relative position
  - preservation of embedding
  - planarity
- soft constraints:

  minimize $\lambda_{\text{bends}} \operatorname{cost}_{\text{bends}} + \lambda_{\text{length}} \operatorname{cost}_{\text{length}} + \lambda_{\text{relpos}} \operatorname{cost}_{\text{relpos}}$

Summary of the MIP

- hard constraints:
  - octilinearity
  - minimum edge length
  - (partially) relative position
  - preservation of embedding
  - planarity
- soft constraints:

  minimize $\lambda_{\text{bends}}\,\text{cost}_{\text{bends}} + \lambda_{\text{length}}\,\text{cost}_{\text{length}} + \lambda_{\text{relpos}}\,\text{cost}_{\text{relpos}}$

- models METROMAPLAYOUT as MIP

Summary of the MIP

- hard constraints:
  - octilinearity
  - minimum edge length
  - (partially) relative position
  - preservation of embedding
  - planarity
- soft constraints:

  minimize $\lambda_{\text{bends}}\,\text{cost}_{\text{bends}} + \lambda_{\text{length}}\,\text{cost}_{\text{length}} + \lambda_{\text{relpos}}\,\text{cost}_{\text{relpos}}$

- models MetroMapLayout as MIP
- in total $O(|V|^2)$ constraints and variables

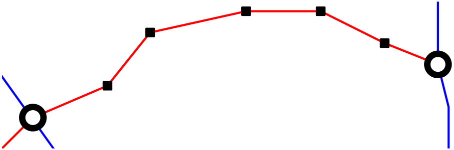
Summary of the MIP

- hard constraints:
  - octilinearity
  - minimum edge length
  - (partially) relative position
  - preservation of embedding
  - planarity
- soft constraints:

  minimize $\lambda_{\text{bends}}\,\text{cost}_{\text{bends}} + \lambda_{\text{length}}\,\text{cost}_{\text{length}} + \lambda_{\text{relpos}}\,\text{cost}_{\text{relpos}}$

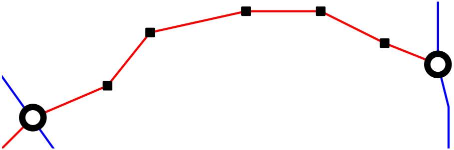
- models METROMAPLAYOUT as MIP
- in total $O(|V|^2)$ constraints and variables

Speed-Up Techniques: Reduce Graph Size



- metro graphs have many degree-2 vertices
- want to optimize line straightness

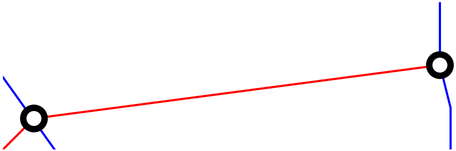
Speed-Up Techniques: Reduce Graph Size



- metro graphs have many degree-2 vertices
- want to optimize line straightness

Idea 1 collapse all degree-2 vertices

Speed-Up Techniques: Reduce Graph Size



- metro graphs have many degree-2 vertices
- want to optimize line straightness

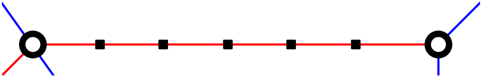
Idea 1 collapse all degree-2 vertices

Speed-Up Techniques: Reduce Graph Size

- metro graphs have many degree-2 vertices
- want to optimize line straightness

Idea 1 collapse all degree-2 vertices

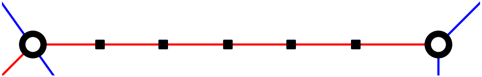
Speed-Up Techniques: Reduce Graph Size



- metro graphs have many degree-2 vertices
- want to optimize line straightness

Idea 1 collapse all degree-2 vertices

Speed-Up Techniques: Reduce Graph Size

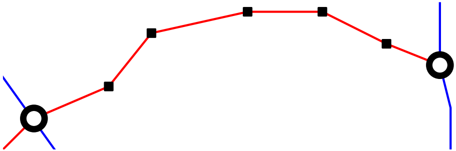


- metro graphs have many degree-2 vertices
- want to optimize line straightness

Idea 1 collapse all degree-2 vertices

- low flexibility

Speed-Up Techniques: Reduce Graph Size



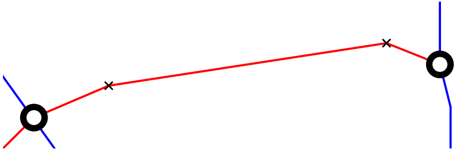
- metro graphs have many degree-2 vertices
- want to optimize line straightness

Idea 1 collapse all degree-2 vertices

- low flexibility

Idea 2 keep two *joints*

Speed-Up Techniques: Reduce Graph Size



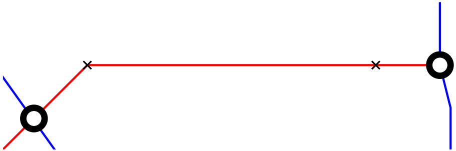
- metro graphs have many degree-2 vertices
- want to optimize line straightness

Idea 1 collapse all degree-2 vertices

- low flexibility

Idea 2 keep two *joints*

Speed-Up Techniques: Reduce Graph Size

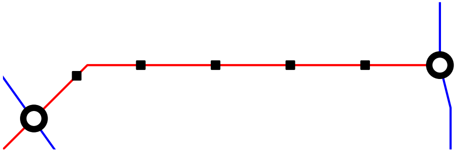


- metro graphs have many degree-2 vertices
- want to optimize line straightness

Idea 1 collapse all degree-2 vertices
- low flexibility

Idea 2 keep two *joints*

Speed-Up Techniques: Reduce Graph Size



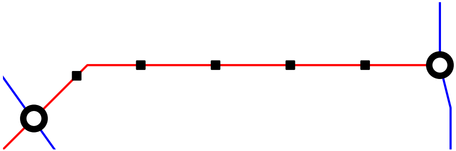
- metro graphs have many degree-2 vertices
- want to optimize line straightness

Idea 1 collapse all degree-2 vertices

- low flexibility

Idea 2 keep two *joints*

Speed-Up Techniques: Reduce Graph Size



- metro graphs have many degree-2 vertices
- want to optimize line straightness

Idea 1 collapse all degree-2 vertices
- low flexibility

Idea 2 keep two *joints*
- higher flexibility
- more similar to input

Speed-Up Techniques: Reduce MIP Size

- $O(|V|^2)$ planarity constraints (for each pair of edges...)
- in practice 95–99% of constraints

Speed-Up Techniques: Reduce MIP Size

- $O(|V|^2)$ planarity constraints (for each pair of edges...)
- in practice 95–99% of constraints

Observation 1

- consider only pairs of edges incident to the same face
- still $O(|V|^2)$ constraints

Speed-Up Techniques: Reduce MIP Size

- $O(|V|^2)$ planarity constraints (for each pair of edges...)
- in practice 95–99% of constraints

Observation 1

- consider only pairs of edges incident to the same face
- still $O(|V|^2)$ constraints

Observation 2

- in practice no or only few crossings due to soft constraints

Speed-Up Techniques: Reduce MIP Size

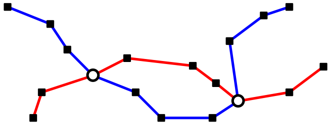
- $O(|V|^2)$ planarity constraints (for each pair of edges...)
- in practice 95–99% of constraints

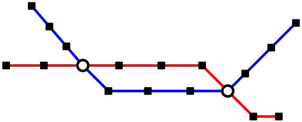
Observation 1

- consider only pairs of edges incident to the same face
- still $O(|V|^2)$ constraints

Observation 2

- in practice no or only few crossings due to soft constraints





Speed-Up Techniques: Reduce MIP Size

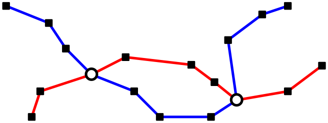
- $O(|V|^2)$ planarity constraints (for each pair of edges...)
- in practice 95–99% of constraints

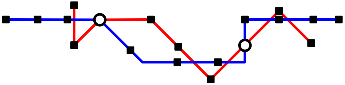
Observation 1

- consider only pairs of edges incident to the same face
- still $O(|V|^2)$ constraints

Observation 2

- in practice no or only few crossings due to soft constraints





Speed-Up Techniques: Callback Functions

- MIP optimizer CPLEX offers advanced callback functions
- add required planarity constraints on the fly

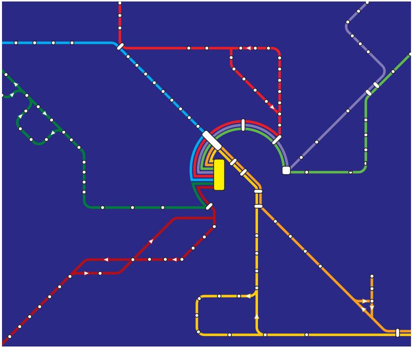
Algorithm

1. start solving MIP without planarity constraints
2. for each new solution
    1. interrupt CPLEX
    2. if solution is not planar
        - add planarity constraints for intersecting edges
        - reject solution

       else
        - accept solution
    3. continue solving the MIP (until optimal)

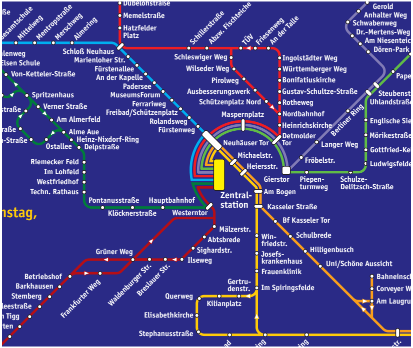
Labeling

- unlabeled metro map of little use in practice



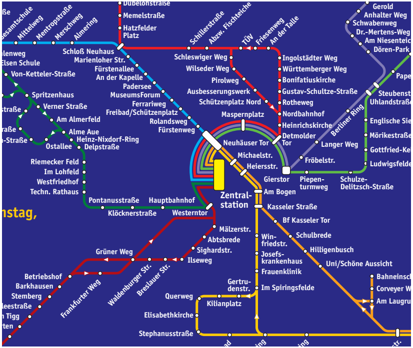
Labeling

- unlabeled metro map of little use in practice
- labels
  - occupy space
  - may not overlap



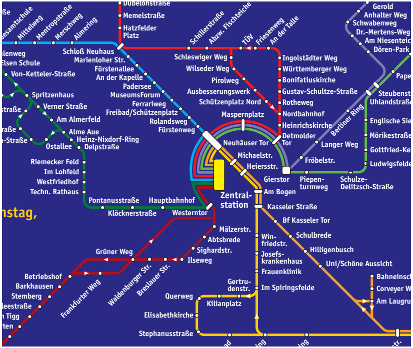
Labeling

- unlabeled metro map of little use in practice
- labels
  - occupy space
  - may not overlap
- static edge labeling is NP-hard [Tollis, Kakoulis '01]



Labeling

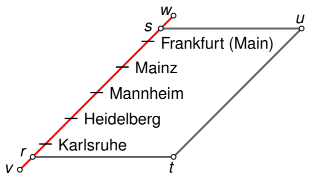
- unlabeled metro map of little use in practice
- labels
  - occupy space
  - may not overlap
- static edge labeling is NP-hard [Tollis, Kakoulis '01]
- combine layout and labeling for better results



Modeling Labels

Model labels as special metro lines:

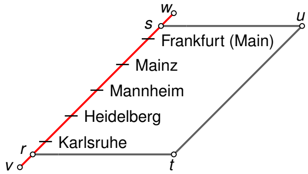
- put all labels between each pair of interchange stations into one parallelogram,



Modeling Labels

Model labels as special metro lines:

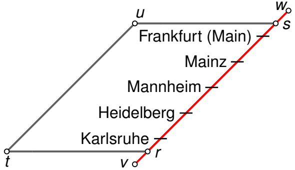
- put all labels between each pair of interchange stations into one parallelogram,
- allow parallelograms to change sides,



Modeling Labels

Model labels as special metro lines:

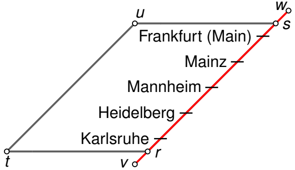
- put all labels between each pair of interchange stations into one parallelogram,
- allow parallelograms to change sides,



Modeling Labels

Model labels as special metro lines:

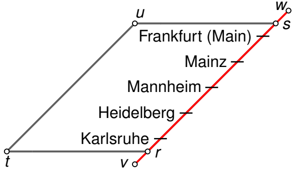
- put all labels between each pair of interchange stations into one parallelogram,
- allow parallelograms to change sides,
- bad news: a **lot** more planarity constraints



Modeling Labels

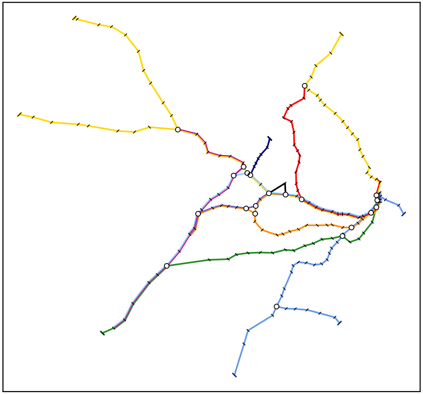
Model labels as special metro lines:

- put all labels between each pair of interchange stations into one parallelogram,
- allow parallelograms to change sides,
- bad news: a **lot** more planarity constraints
- good news: callback method helps



Results – Sydney unlabeled

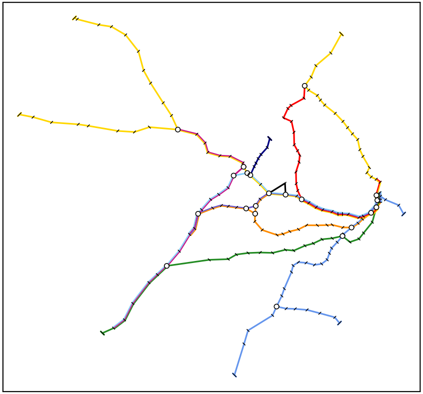
Input



Input	$\lvert V \rvert$	$\lvert E \rvert$	fcs.	$\lvert \mathcal{L} \rvert$
full	174	183	11	10
reduced	88	97		

Results – Sydney unlabeled

Input



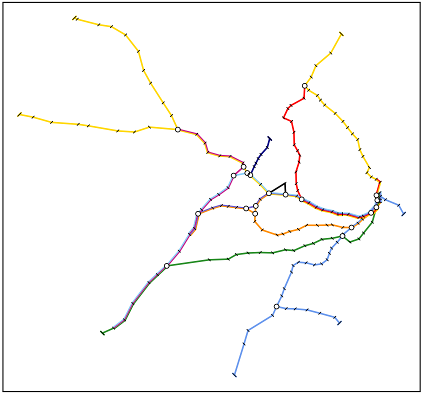
Input	$\|V\|$	$\|E\|$	fcs.	$\|\mathcal{L}\|$
full	174	183	11	10
reduced	88	97		

↓

MIP	constr.	var.
full	152,194	37,802
callback	3,529	4,834
skipped	3,034	1,642

Results – Sydney unlabeled

Input



Input	$\|V\|$	$\|E\|$	fcs.	$\|\mathcal{L}\|$
full	174	183	11	10
reduced	88	97		

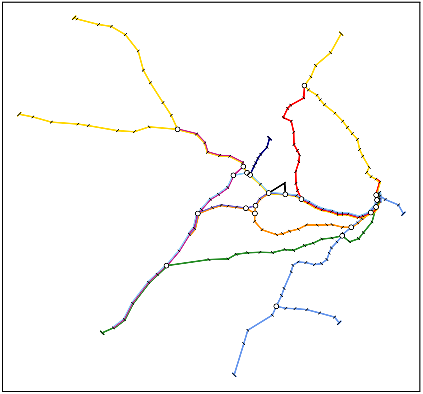
$\downarrow$

MIP	constr.	var.
full	152,194	37,802
callback*	3,529	4,834
skipped	3,034	1,642

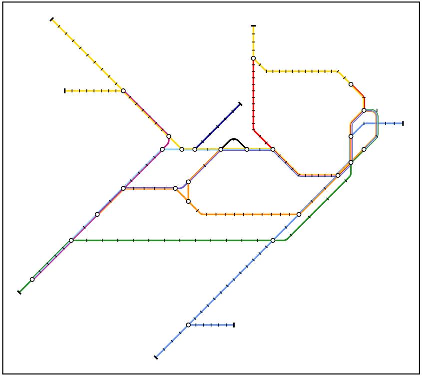
*) 23 minutes w/o proof of opt.
constr. of 3 edge pairs added

Results – Sydney unlabeled

Input

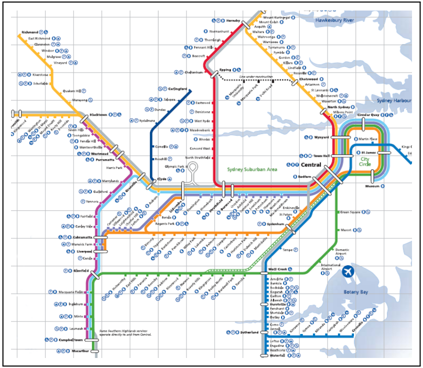


Output (23 min.)

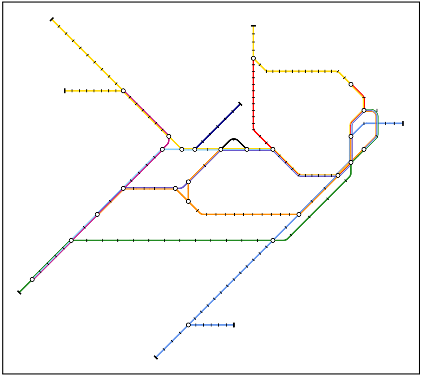


Results – Sydney unlabeled

Official map

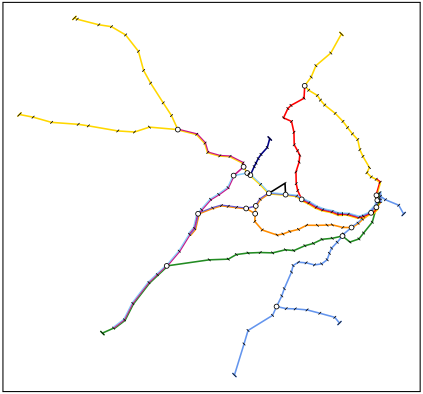


Output (23 min.)



Results – Sydney labeled

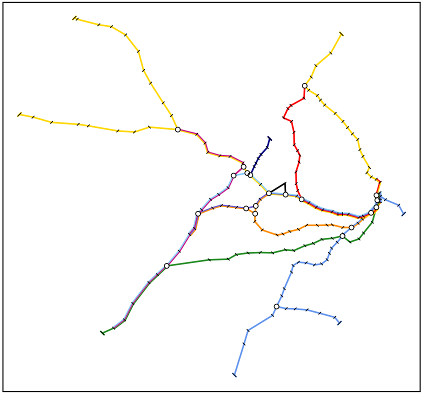
Input



Input	$\|V\|$	$\|E\|$	fcs.	$\|\mathcal{L}\|$
full	174	183	11	10
reduced	88	97		
labeled	242	270	30	

Results – Sydney labeled

Input



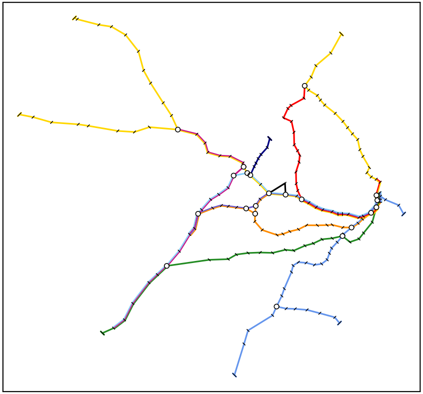
Input	$\lvert V\rvert$	$\lvert E\rvert$	fcs.	$\lvert\mathcal{L}\rvert$
full	174	183	11	10
reduced	88	97		
labeled	242	270	30	

↓

MIP	constr.	var.
full	1,191,406	290,137
callback	21,988	92,681
skipped	6,838	2,969

Results – Sydney labeled

Input



Input	$\|V\|$	$\|E\|$	fcs.	$\|\mathcal{L}\|$
full	174	183	11	
reduced	88	97		10
labeled	242	270	30	

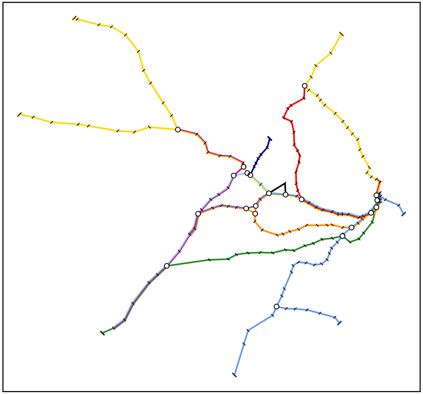
↓

MIP	constr.	var.
full	1,191,406	290,137
callback*	21,988	92,681
skipped	6,838	2,969

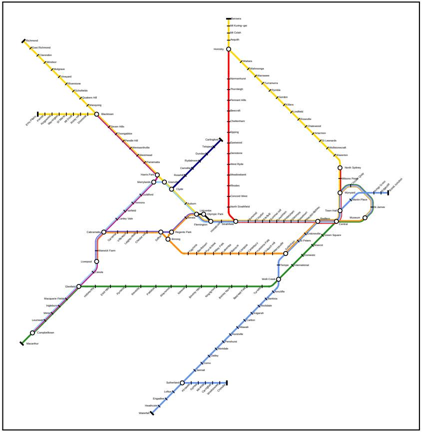
*) 10:30 hours w/o proof of opt.
add constr. of 123 edge pairs

Results – Sydney labeled

Input

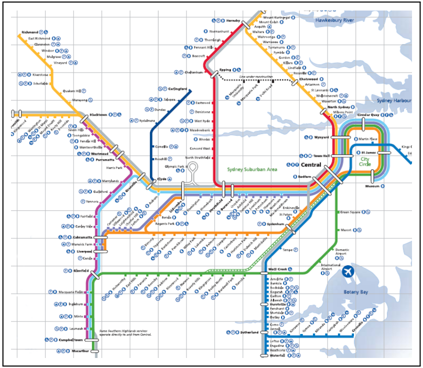


Output (10:30 hrs.)

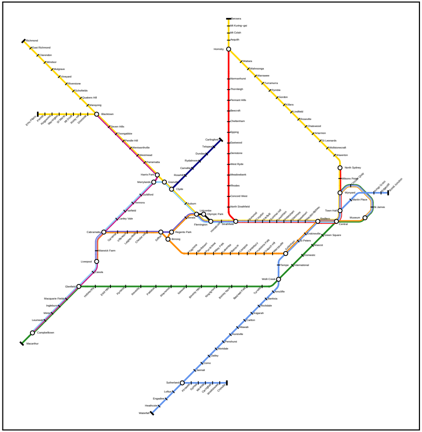


Results – Sydney labeled

Official map

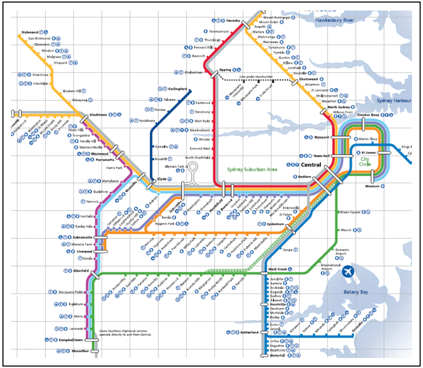


Output (10:30 hrs.)

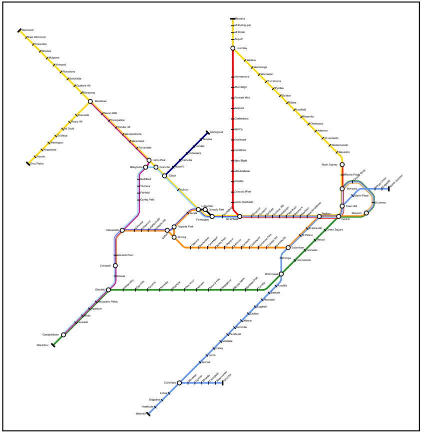


Results – Sydney labeled

Official map

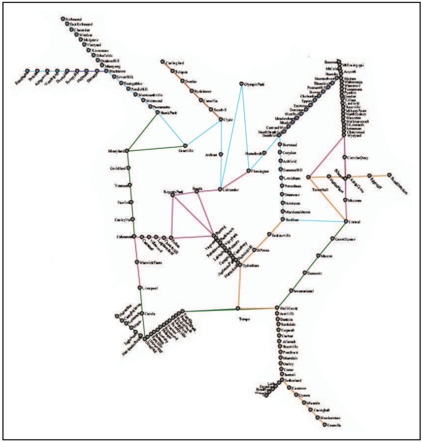


Output (1:40 hrs.)

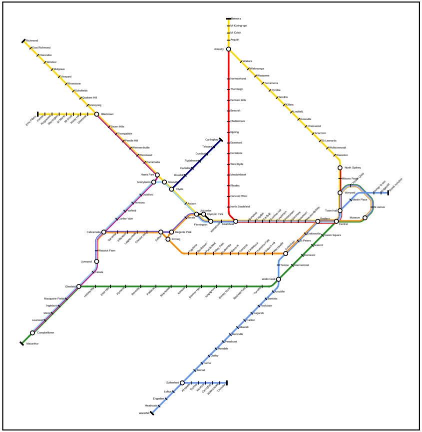


Sydney: Related Work

[Hong et al. GD'04] (7.6 sec.)

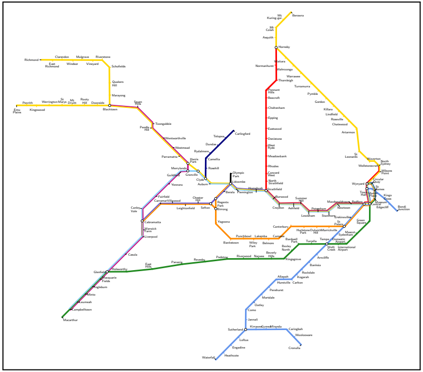


Our output (10:30 hrs.)

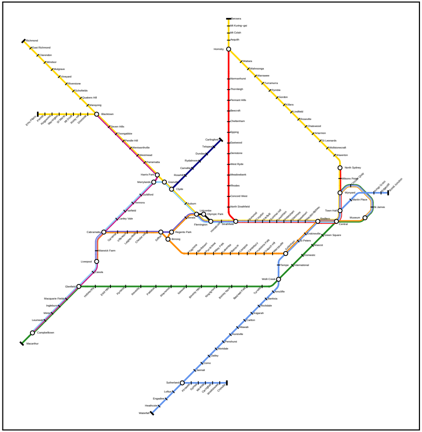


Sydney: Related Work

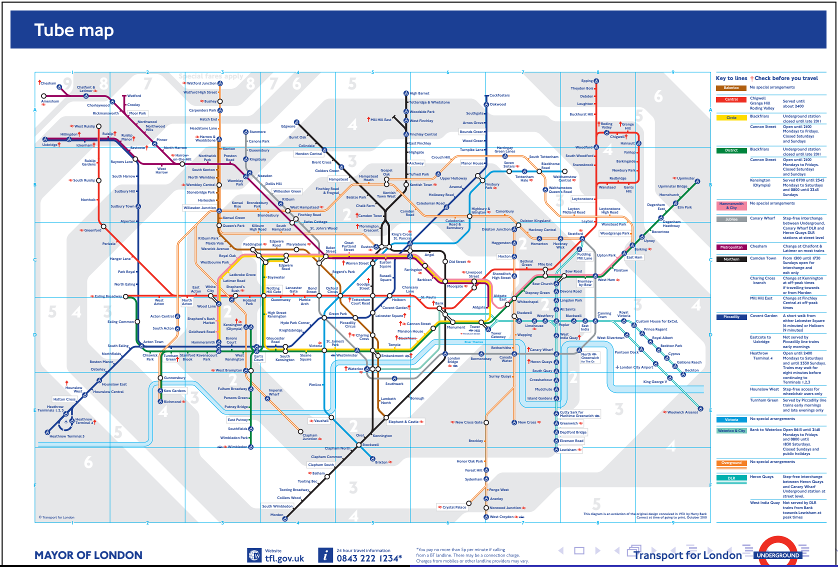
[Stott, Rodgers TVCG'10] (2 hrs.)



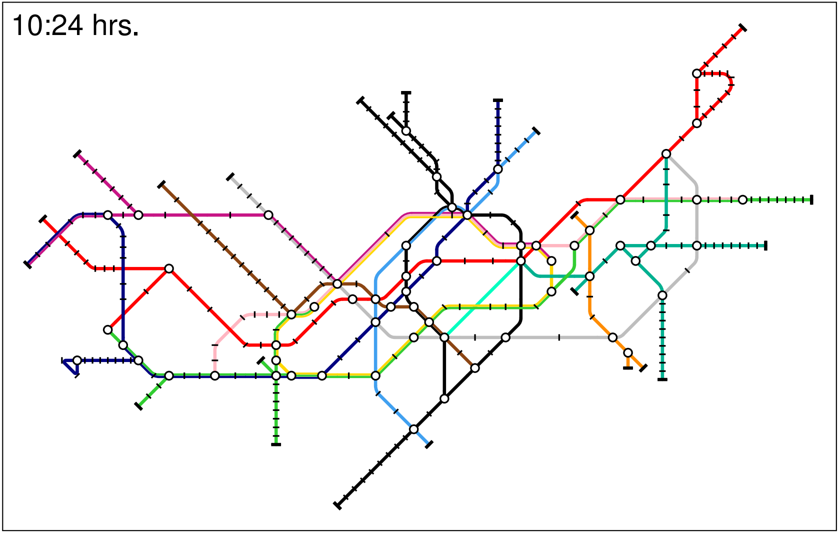
Our output (10:30 hrs.)



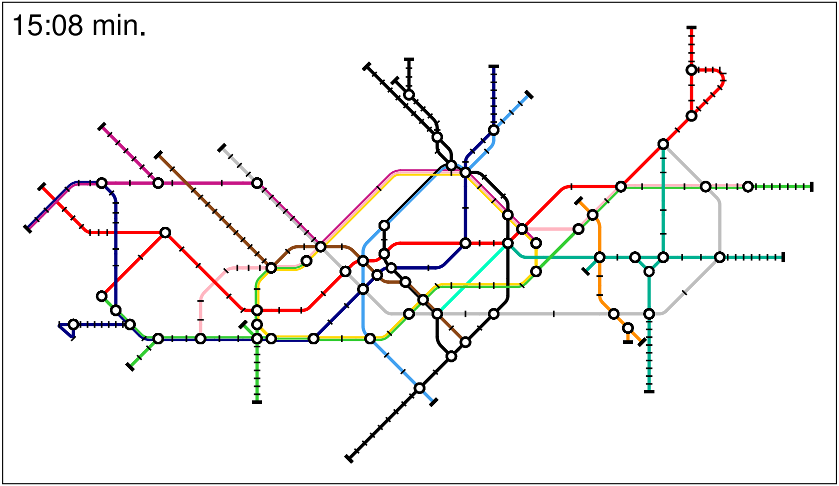
Large Example: London



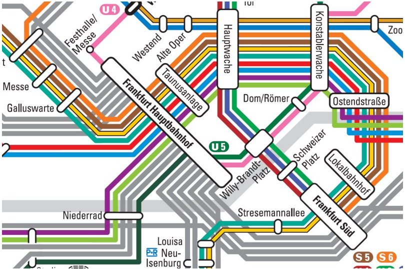
Large Example: London

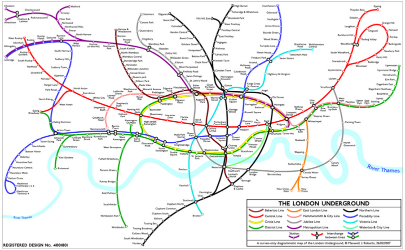


Large Example: London



Are we done yet?





- more user interaction
- how to handle large stations and many parallel lines?
- formulate global aesthetics like symmetry and balance
- octilinear maps are not necessarily always best – how to compute curved metro maps?

Summary

- METROMAPLAYOUT is NP-hard
- formulation of hard and soft constraints as MIP
- combined layout and labeling
- MIP size & runtime reductions
- high-quality results
- MIP can schematize *any* kind of graph sketch

Summary

- METROMAPLAYOUT is NP-hard
- formulation of hard and soft constraints as MIP
- combined layout and labeling
- MIP size & runtime reductions
- high-quality results
- MIP can schematize *any* kind of graph sketch

For more info see:

M. Nöllenburg and A. Wolff. *Drawing and labeling high-quality metro maps by mixed-integer programming.* IEEE Trans. Visualization and Computer Graphics 17(5):626–641, 2011.