Order in the Underground –
How to Automate the Drawing of Metro Maps

Martin Nöllenburg and Alexander Wolff

Department of Computer Science
Karlsruhe University

GD 2005
Outline

1. Modeling the Metro Map Problem
   - What is a metro map?
   - Hard and soft constraints

2. Our Solution
   - Mixed-integer programming formulation
   - Experiments
   - Labeling

3. NP-Hardness
   - Rectilinear vs. octilinear drawing
   - Reduction from planar 3-SAT
Outline

1. Modeling the Metro Map Problem
   - What is a metro map?
   - Hard and soft constraints

2. Our Solution
   - Mixed-integer programming formulation
   - Experiments
   - Labeling

3. NP-Hardness
   - Rectilinear vs. octilinear drawing
   - Reduction from planar 3-SAT
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 ↔ abstract graph
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 $G = (V, E)$, $V \subseteq \mathbb{R}^2$, 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?
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
More Formally

The Metro Map Problem

Given: planar embedded graph $G = (V, E)$, $V \subseteq \mathbb{R}^2$, 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,
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.
(H1) preserve embedding of $G$
Hard Constraints

(H1) preserve embedding of $G$

(H2) draw all edges as octilinear line segments, i.e., parallel to a coordinate axes or at 45° degrees
Hard Constraints

(H1) preserve embedding of $G$

(H2) draw all edges as octilinear line segments, i.e., parallel to a coordinate axes or at $45^\circ$ 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., parallel to a coordinate axes or at 45° degrees
(H3) draw each edge $e$ with length $\geq \ell_e$
(H4) keep vertices $d_{\text{min}}$ away from non-incident edges
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 its relative position
Outline

1. Modeling the Metro Map Problem
   - What is a metro map?
   - Hard and soft constraints

2. Our Solution
   - Mixed-integer programming formulation
   - Experiments
   - Labeling

3. NP-Hardness
   - Rectilinear vs. octilinear drawing
   - Reduction from planar 3-SAT
Mathematical Programming

- **Linear Programming**: efficient optimization method for
  - linear constraints and objective function,
  - real-valued variables (domain \( \mathbb{R} \)).
Mathematical Programming

- **Linear Programming**: efficient optimization method for
  - linear constraints and objective function,
  - real-valued variables (domain \( \mathbb{R} \)).

- **Mixed-Integer Programming (MIP)**:
  - allows also integer variables (domain \( \mathbb{Z} \)),
  - solution NP-hard in general.
Mathematical Programming

- **Linear Programming**: efficient optimization method for
  - linear constraints and objective function,
  - real-valued variables (domain $\mathbb{R}$).
- **Mixed-Integer Programming (MIP)**:
  - allows also integer variables (domain $\mathbb{Z}$),
  - solution NP-hard in general.
- Still a practical method for many hard optimization problems.
Mathematical Programming

- **Linear Programming**: efficient optimization method for
  - linear constraints and objective function,
  - real-valued variables (domain $\mathbb{R}$).
- **Mixed-Integer Programming (MIP)**:
  - allows also integer variables (domain $\mathbb{Z}$),
  - solution NP-hard in general.

- Still a practical method for many hard optimization problems.

---

**Theorem (Nöllenburg & Wolff GD’05)**

The problem MetroMapLayout can be formulated as a MIP s.th.

- linear constraints $\rightarrow$ hard constraints,
- objective function $\rightarrow$ soft constraints.
Example: Octilinearity and Relative Position
Example: Octilinearity and Relative Position

Sectors

- for each vtx. $u$ partition plane into sectors 0–7
  - here: $\text{sec}(u, v) = 5$ (input)
Example: Octilinearity and Relative Position

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)
Example: Octilinearity and Relative Position

Sectors
- for each vtx. $u$ partition plane into sectors 0–7
  - here: $sec(u, v) = 5$ (input)
- number octilinear edge directions accordingly
  - e.g., $dir(u, v) = 4$ (output)

Coordinates
assign $z_1$- and $z_2$-coordinates to each vertex $v$:
- $z_1(v) = x(v) + y(v)$
- $z_2(v) = x(v) - y(v)$
Example: Octilinearity and Relative Position

Goal

Draw edge $uv$
- with minimum length $\ell_{uv}$
- restricted to 3 directions
Example: Octilinearity and Relative Position

Goal

Draw edge $uv$
- with minimum length $\ell_{uv}$
- restricted to 3 directions

How to model this using linear constraints?
Example: Octilinearity and Relative Position

Goal

Draw edge $uv$
- with minimum length $\ell_{uv}$
- restricted to 3 directions

How to model this using linear constraints?

Binary Variables

$$\alpha_{\text{prev}}(u, v) + \alpha_{\text{orig}}(u, v) + \alpha_{\text{next}}(u, v) = 1$$
Example: Octilinearity and Relative Position

\[ \begin{align*}
  y(u) - y(v) & \leq M(1 - \alpha_{\text{prev}}(u, v)) \\
  -y(u) + y(v) & \leq M(1 - \alpha_{\text{prev}}(u, v)) \\
  x(u) - x(v) & \geq -M(1 - \alpha_{\text{prev}}(u, v)) + \ell_{uv}
\end{align*} \]
Example: Octilinearity and Relative Position

Previous Sector

\[ \begin{align*}
  y(u) - y(v) & \leq M(1 - \alpha_{\text{prev}}(u, v)) \\
  -y(u) + y(v) & \leq M(1 - \alpha_{\text{prev}}(u, v)) \\
  x(u) - x(v) & \geq -M(1 - \alpha_{\text{prev}}(u, v)) + \ell_{uv}
\end{align*} \]

How does this work?
Example: Octilinearity and Relative Position

Previous Sector

\[
\begin{align*}
  y(u) - y(v) & \leq M(1 - \alpha_{\text{prev}}(u, v)) \\
  -y(u) + y(v) & \leq M(1 - \alpha_{\text{prev}}(u, v)) \\
  x(u) - x(v) & \geq -M(1 - \alpha_{\text{prev}}(u, v)) + \ell_{uv}
\end{align*}
\]

How does this work?

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

\[
\begin{align*}
  y(u) - y(v) & \leq M \\
  -y(u) + y(v) & \leq M \\
  x(u) - x(v) & \geq \ell_{uv} - M
\end{align*}
\]
Example: Octilinearity and Relative Position

Previous Sector

\[
\begin{align*}
y(u) - y(v) & \leq M(1 - \alpha_{\text{prev}}(u, v)) \\
-y(u) + y(v) & \leq M(1 - \alpha_{\text{prev}}(u, v)) \\
x(u) - x(v) & \geq -M(1 - \alpha_{\text{prev}}(u, v)) + \ell_{uv}
\end{align*}
\]

How does this work?

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

\[
\begin{align*}
y(u) - y(v) & \leq 0 \\
-y(u) + y(v) & \leq 0 \\
x(u) - x(v) & \geq \ell_{uv}
\end{align*}
\]
Example: Octilinearity and Relative Position

Original Sector

\[
\begin{align*}
    z_2(u) - z_2(v) & \leq M(1 - \alpha_{\text{orig}}(u, v)) \\
    -z_2(u) + z_2(v) & \leq M(1 - \alpha_{\text{orig}}(u, v)) \\
    z_1(u) - z_1(v) & \geq -M(1 - \alpha_{\text{orig}}(u, v)) + 2\ell_{uv}
\end{align*}
\]
Example: Octilinearity and Relative Position

Original Sector

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

Next Sector

\[
\begin{align*}
x(u) - x(v) & \leq M(1 - \alpha_{\text{next}}(u, v)) \\
-x(u) + x(v) & \leq M(1 - \alpha_{\text{next}}(u, v)) \\
y(u) - y(v) & \geq -M(1 - \alpha_{\text{next}}(u, v)) + \ell_{uv}
\end{align*}
\]
Summary of the Model

- The above constraints enforce
  - octilinearity,
  - minimum edge length,
  - (partially) relative position
Summary of the Model

- The above constraints enforce
  - octilinearity,
  - minimum edge length,
  - (partially) relative position

- Similarly:
  - preservation of embedding
  - planarity
Summary of the Model

- The above constraints enforce
  - octilinearity,
  - minimum edge length,
  - (partially) relative position

- Similarly:
  - preservation of embedding
  - planarity

- Soft constraints: weighted sum in objective function

\[
\text{minimize } \lambda_{\text{bends}} \text{cost}_{\text{bends}} + \lambda_{\text{length}} \text{cost}_{\text{length}} + \lambda_{\text{dir}} \text{cost}_{\text{dir}}
\]
Summary of the Model

- The above constraints enforce
  - octilinearity,
  - minimum edge length,
  - (partially) relative position

- Similarly:
  - preservation of embedding
  - planarity

- Soft constraints: weighted sum in objective function
  \[
  \text{minimize } \lambda_{\text{bends}} \text{cost}_{\text{bends}} + \lambda_{\text{length}} \text{cost}_{\text{length}} + \lambda_{\text{dir}} \text{cost}_{\text{dir}}
  \]

- In total $O(|V|^2)$ constraints and variables
Summary of the Model

- The above constraints enforce
  - octilinearity,
  - minimum edge length,
  - (partially) relative position
- Similarly:
  - preservation of embedding
  - planarity
- Soft constraints: weighted sum in objective function
  \[
  \text{minimize } \lambda_{\text{bends}} \text{cost}_{\text{bends}} + \lambda_{\text{length}} \text{cost}_{\text{length}} + \lambda_{\text{dir}} \text{cost}_{\text{dir}}
  \]
- In total \(O(|V|^2)\) constraints and variables
Results – Vienna

| Input          | \( |V| \) | \( |E| \) | lines |
|----------------|--------|--------|-------|
| normal         | 90     | 96     | 5     |
| reduced        | 44     | 50     |       |
Results – Vienna

Input

<table>
<thead>
<tr>
<th>Input</th>
<th>V</th>
<th>E</th>
<th>lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>90</td>
<td>96</td>
<td>5</td>
</tr>
<tr>
<td>reduced</td>
<td>44</td>
<td>50</td>
<td></td>
</tr>
</tbody>
</table>

↓

MIP

<table>
<thead>
<tr>
<th></th>
<th>constr.</th>
<th>var.</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>39363</td>
<td>9960</td>
</tr>
<tr>
<td>improved</td>
<td>23226</td>
<td>6048</td>
</tr>
<tr>
<td>heuristic 1</td>
<td>5703</td>
<td>1800</td>
</tr>
<tr>
<td>heuristic 2</td>
<td>1875</td>
<td>872</td>
</tr>
</tbody>
</table>
Results – Vienna

<table>
<thead>
<tr>
<th>Input</th>
<th>$V$</th>
<th>$E$</th>
<th>lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>90</td>
<td>96</td>
<td>5</td>
</tr>
<tr>
<td>reduced</td>
<td>44</td>
<td>50</td>
<td></td>
</tr>
</tbody>
</table>

\[
\Downarrow
\]

<table>
<thead>
<tr>
<th>MIP</th>
<th>constr.</th>
<th>var.</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>39363</td>
<td>9960</td>
</tr>
<tr>
<td>improved</td>
<td>23226</td>
<td>6048</td>
</tr>
<tr>
<td>heuristic 1</td>
<td>5703</td>
<td>1800</td>
</tr>
<tr>
<td>heuristic 2*</td>
<td>1875</td>
<td>872</td>
</tr>
</tbody>
</table>

*) 29 seconds w/o proof of opt.
Results – Vienna

Input

Output
Results – Vienna

Official map

Output
Results – Sydney

<table>
<thead>
<tr>
<th>Input</th>
<th>( V )</th>
<th>( E )</th>
<th>lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>174</td>
<td>183</td>
<td>10</td>
</tr>
<tr>
<td>reduced</td>
<td>62</td>
<td>71</td>
<td></td>
</tr>
</tbody>
</table>
Results – Sydney

### Input

<table>
<thead>
<tr>
<th></th>
<th>V</th>
<th>E</th>
<th>lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>174</td>
<td>183</td>
<td>10</td>
</tr>
<tr>
<td>reduced</td>
<td>62</td>
<td>71</td>
<td></td>
</tr>
</tbody>
</table>

↓

### MIP

<table>
<thead>
<tr>
<th></th>
<th>constr.</th>
<th>var.</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>81416</td>
<td>20329</td>
</tr>
<tr>
<td>improved</td>
<td>45182</td>
<td>11545</td>
</tr>
<tr>
<td>heuristic 1</td>
<td>6242</td>
<td>2105</td>
</tr>
<tr>
<td>heuristic 2</td>
<td>3041</td>
<td>1329</td>
</tr>
</tbody>
</table>
Results – Sydney

\begin{tabular}{|l|c|c|c|}
\hline
\textbf{Input} & \textbf{\(|V|\)} & \textbf{\(|E|\)} & \textbf{lines} \\
\hline
\textbf{normal} & 174 & 183 & 10 \\
\textbf{reduced} & 62 & 71 & \\
\hline
\end{tabular}

\begin{tabular}{|l|c|c|}
\hline
\textbf{MIP} & \textbf{constr.} & \textbf{var.} \\
\hline
\textbf{normal} & 81416 & 20329 \\
\textbf{improved} & 45182 & 11545 \\
\textbf{heuristic 1⋆} & 6242 & 2105 \\
\textbf{heuristic 2} & 3041 & 1329 \\
\hline
\end{tabular}

⋆) 22 minutes w/o proof of opt.
Results – Sydney

Input

Output
Results – Sydney

Official map

Output
Sydney: Related Work

[Hong et al.: GD04] 7.6 sec.

Output (22 min.)
Sydney: Related Work

[Stott & Rodgers: IV04] reduced: 4 min.  

Output (22 min.)
Sydney: Related Work

[Stott & Rodgers: IV04] normal: 28 min. Output (22 min.)
Labeling

Model labels as special metro lines:
Model labels as special metro lines: put all labels between a pair of interchange stations into one parallelogram,
Model labels as special metro lines:

- put all labels between a pair of interchange stations into one parallelogram,
- allow parallelograms to change sides,
Model labels as special metro lines:
- put all labels between a pair of interchange stations into one parallelogram,
- allow parallelograms to change sides,
- problem: a lot more planarity constraints :-(

Martin Nöllenburg and Alexander Wolff
Our Model

Our Solution

NP-Hardness

Mixed-integer programming formulation

Experiments

Labeling

Labeling

Solution:

solve MIP without planarity constraints
Our Model

Our Solution

NP-Hardness

Mixed-integer programming formulation

Experiments

Labeling

Solution:

- solve MIP without planarity constraints
- use CPLEX callback fct.

Martin Nöllenburg and Alexander Wolff

Drawing Metro Maps
Labeling

Solution:
- solve MIP without planarity constraints
- use CPLEX callback fct.
- test which edge pairs cross
Our Model
Our Solution
NP-Hardness
Mixed-integer programming formulation
Experiments
Labeling

Labeling

Solution:
- solve MIP without planarity constraints
- use CPLEX callback function
- test which edge pairs cross
- add corresponding constraints
**Our Model**

**Our Solution**

**Experiments**

**Labeling**

**NP-Hardness**

Mixed-integer programming formulation

- **Labeling**

- Georges-Vanier
- Lucien L’Allier
- Bonaventure
- Square-Victoria
- Place d’Armes
- Champ-de-Mars
- Snowdon
- Lionel Groulx
- Place-Saint-Henri
- Vendome
- Saint-Laurent
- Place-des-Arts
- Sherbrooke
- Prefontaine
- Frontenac
- Pie-IX
- Joliette
- Prefontaine
- Place-Saint-Henri
- Ville-Marie
- Côte-Vertu
- Côte-des-Neiges
- Côte-Sainte-Catherine
- De La Savane
- Du College
- Atwater
- Jolicoeur
- Angrignon
- De Castelnau
- Frontenac
- Berri-UQAM
- Rosemont
- Côte-Vertu
- Fabre
- Jarry
- Jean-Talon
- Beaudry
- Beaubien
- De Castelnau
- Parc
- Acadie
- Outremont
- Sauve
- Cremazie
- Fabre
- Jarry
- Jean-Talon
- Beaudry
- Beaubien

**Solution:**

- solve MIP without planarity constraints
- use CPLEX callback fct.
- test which edge pairs cross
- add corresponding constr.
- continue to solve MIP
Our Model
Our Solution
NP-Hardness
Mixed-integer programming formulation
Experiments
Labeling

Labeling

Solution:
- solve MIP without planarity constraints
- use CPLEX callback fct.
- test which edge pairs cross
- add corresponding constr.
- continue to solve MIP

Montréal: 17 min.
Labeling

Wien: 1 day.
Labeling

Tokyo: < 10 sec.

[Kameda & Imai: IEICE’03]
Outline

1. Modeling the Metro Map Problem
   - What is a metro map?
   - Hard and soft constraints

2. Our Solution
   - Mixed-integer programming formulation
   - Experiments
   - Labeling

3. NP-Hardness
   - Rectilinear vs. octilinear drawing
   - Reduction from planar 3-SAT
Another 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?
Another 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 ’87)**

PEED can be solved efficiently.
Another 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 ’87)**

RECTILINEARGRAPHDRAWING *can be solved efficiently.*

**Proof.**

By reduction to a flow problem.
Another 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 ’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 (Nöllenburg Master’s Thesis’05)**

METROMAPLAYOUT is NP-hard.

**Proof.**
Reduction from PLANAR 3-SAT to METROMAPLAYOUT.
Outline of the Reduction

Input: planar 3-SAT formula \( \varphi = (x_1 \lor x_3 \lor x_4) \land (x_1 \lor \overline{x_2} \lor \overline{x_3}) \land \ldots \)
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 $G_\varphi$ with:
$G_\varphi$ has a metro map drawing $\iff \varphi$ 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 \( G_\varphi \) with:
\( G_\varphi \) has a metro map drawing \( \iff \varphi \) satisfiable.
Variable Gadget

\[ x = \text{true} \]
Variable Gadget

\[ x = \text{false} \]
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 $G_\varphi$ with:
$G_\varphi$ has a metro map drawing $\iff \varphi$ satisfiable.
Clause Gadget
Clause Gadget
Clause Gadget
Clause Gadget
Clause Gadget
Clause Gadget
Other applications
Clipping

Our Model
Our Solution
NP-Hardness
Rectilinear vs. octilinear drawing
Reduction from planar 3-SAT

Drawing Metro Maps
To do: rectangular stations & multi-edges
Summary (metro maps)

- **METROMAPLAYOUT** is NP-hard.
- Formulated and implemented MIP.
- Our MIP can draw *any* kind of sketch “nicely”.
- Results comparable to manually designed maps.
- Reduced MIP size & runtime drastically.
Summary (metro maps)

- **METROMAPLAYOUT** is NP-hard.
- Formulated and implemented MIP.
- Our MIP can draw *any* kind of sketch “nicely”.
- Results comparable to manually designed maps.
- Reduced MIP size & runtime drastically.

To Do

- rectangular stations
- multi-edges
- user interaction (e.g., fixing certain edge directions)