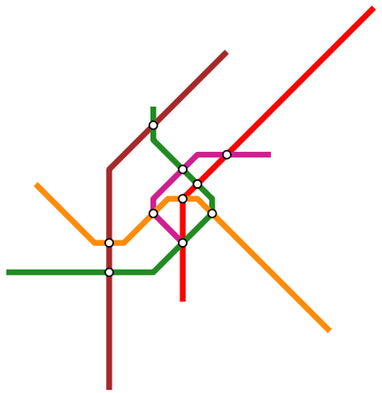
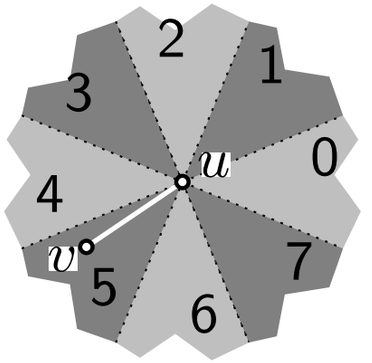
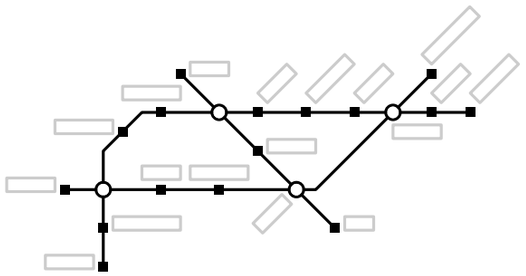
Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Visualization of Graphs



Lecture 12:
Octilinear Graph Drawing
Metro Map Layout

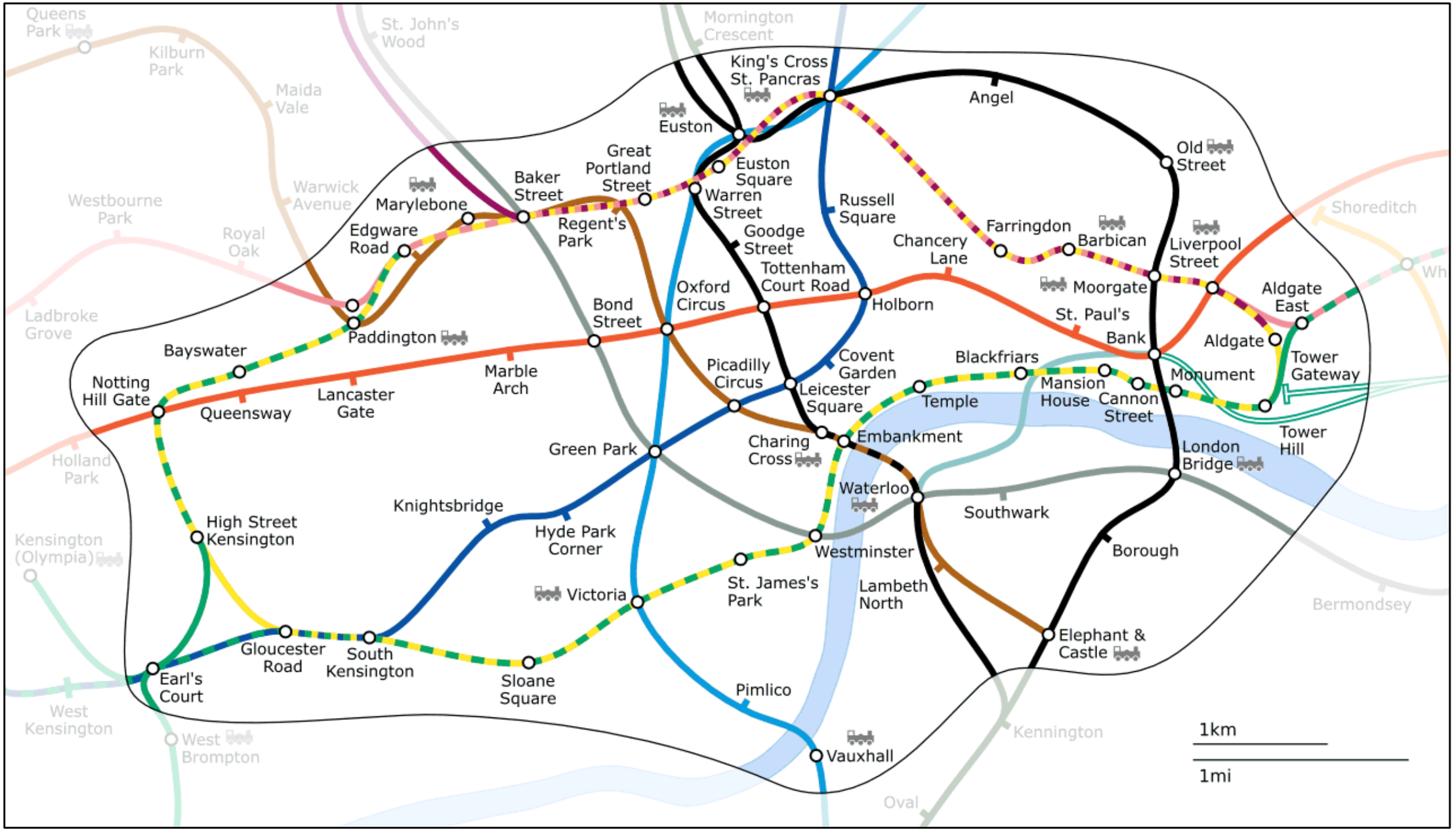




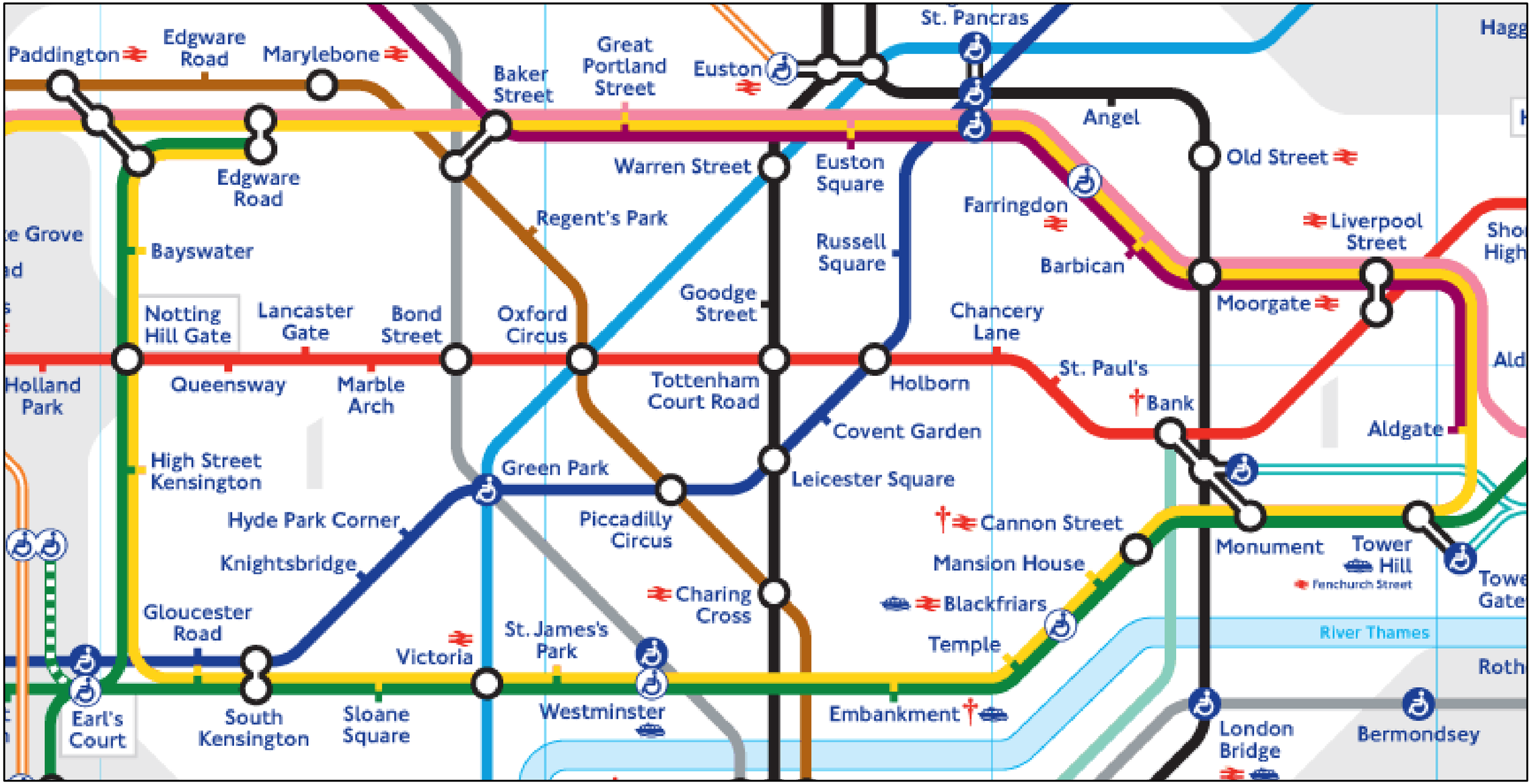
Part I:
Schematic Metro Maps

Alexander Wolff

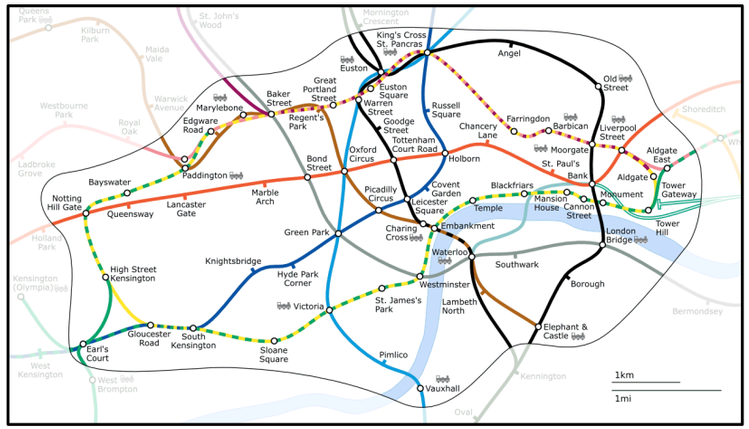
What is a Schematic Metro Map?

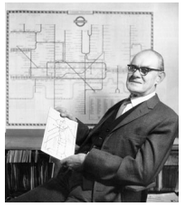


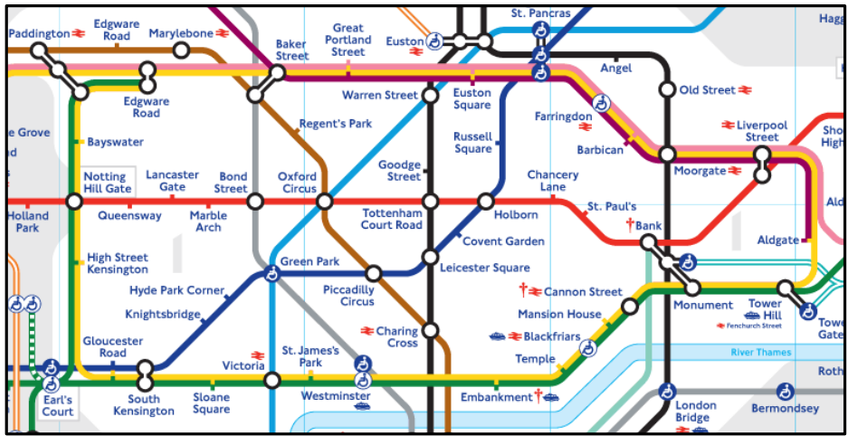
What is a Schematic Metro Map?



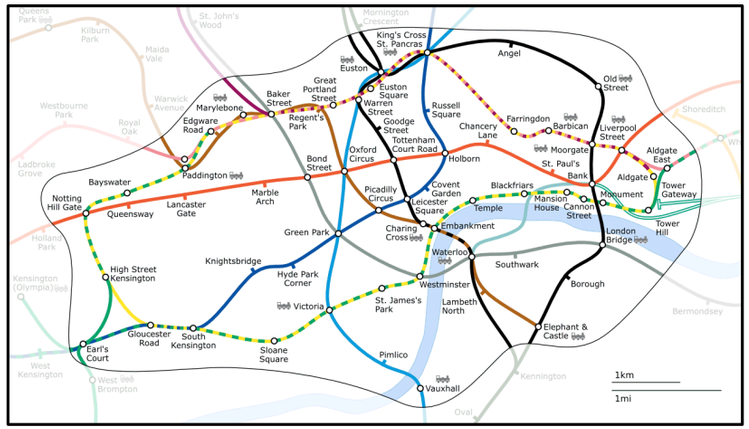
What is a Schematic Metro Map?

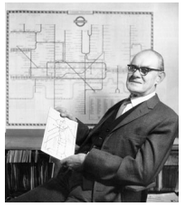


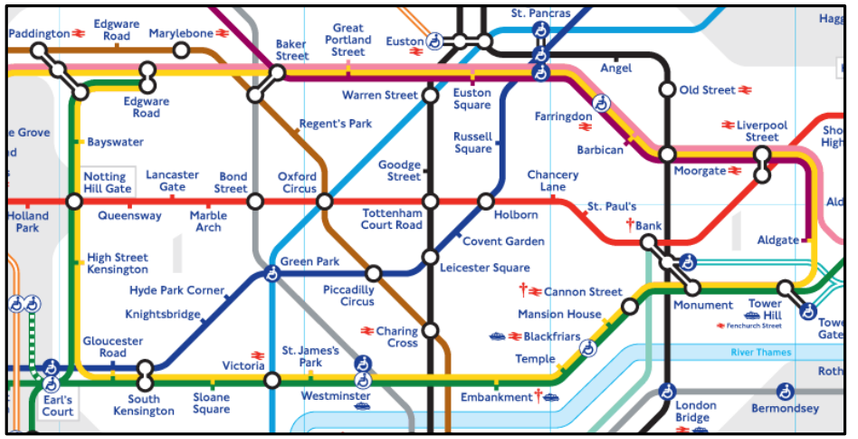




What is a Schematic Metro Map?

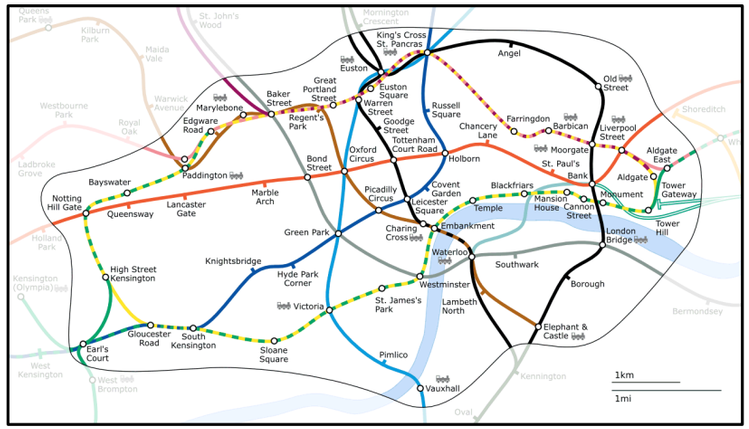




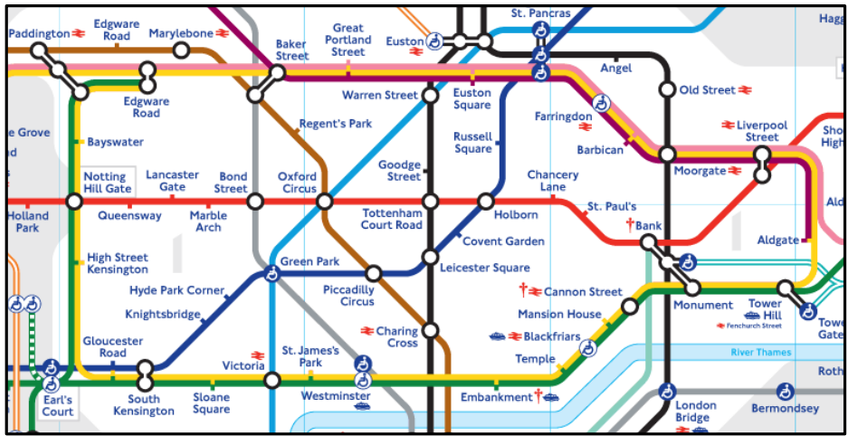


- map/diagram that shows stations connected by metro lines

What is a Schematic Metro Map?

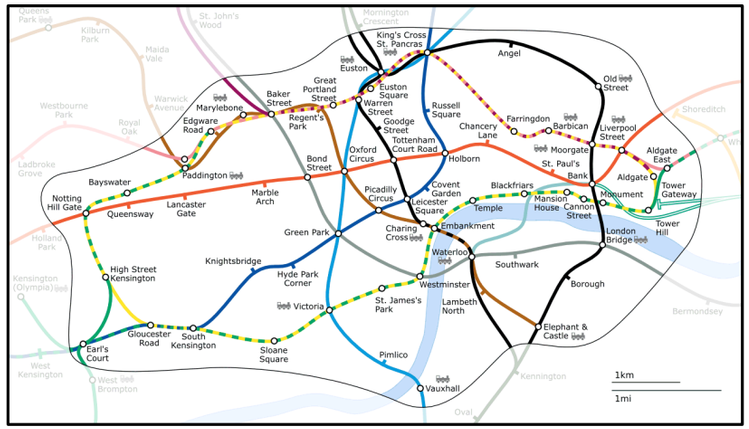




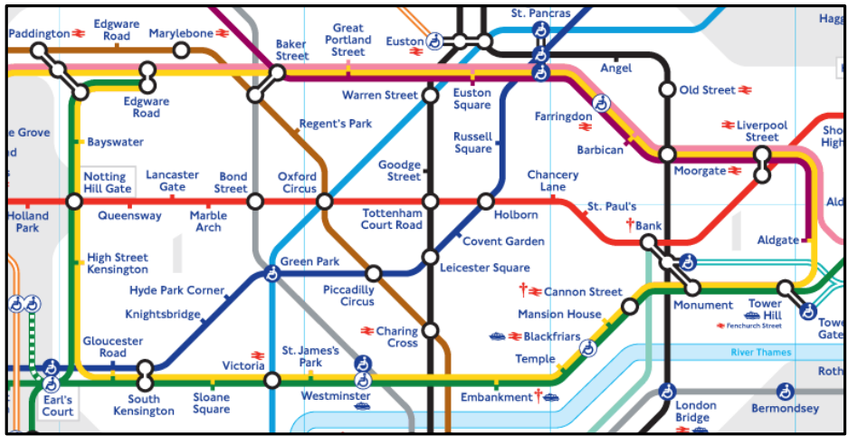


- map/diagram that shows stations connected by metro lines
- focus on topology rather than geography

What is a Schematic Metro Map?

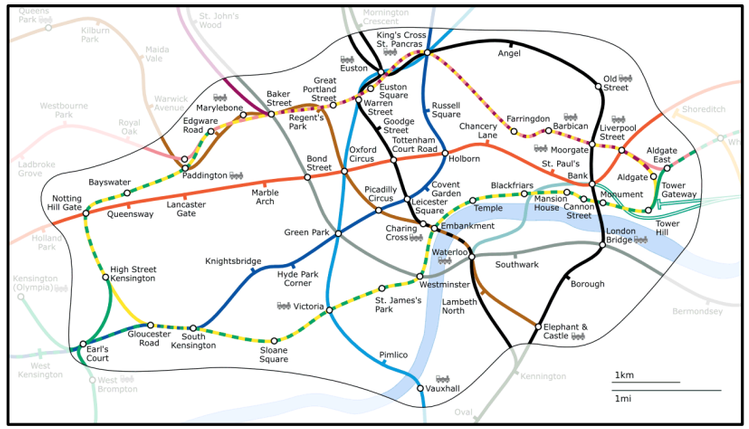


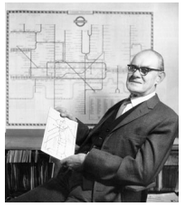


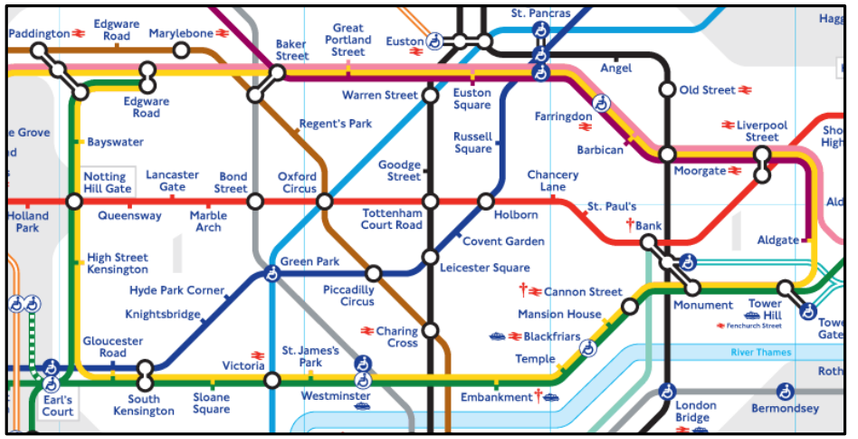


- map/diagram that shows stations connected by metro lines
- focus on topology rather than geography
- goal: easy-to-use visual navigation aid for passengers

What is a Schematic Metro Map?

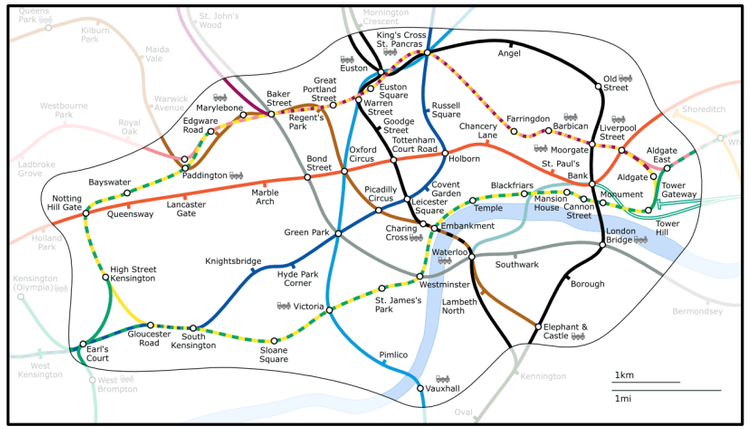


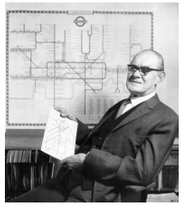


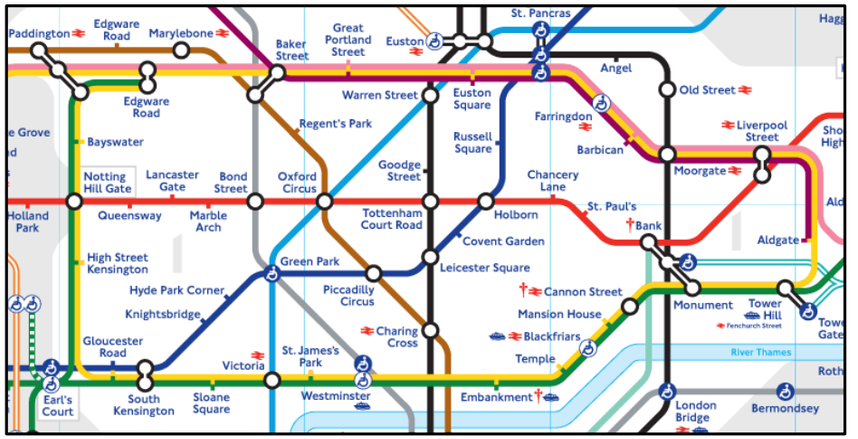


- map/diagram that shows stations connected by metro lines
- focus on topology rather than geography
- goal: easy-to-use visual navigation aid for passengers
  - "*How do I quickly get from A to B?*"

What is a Schematic Metro Map?

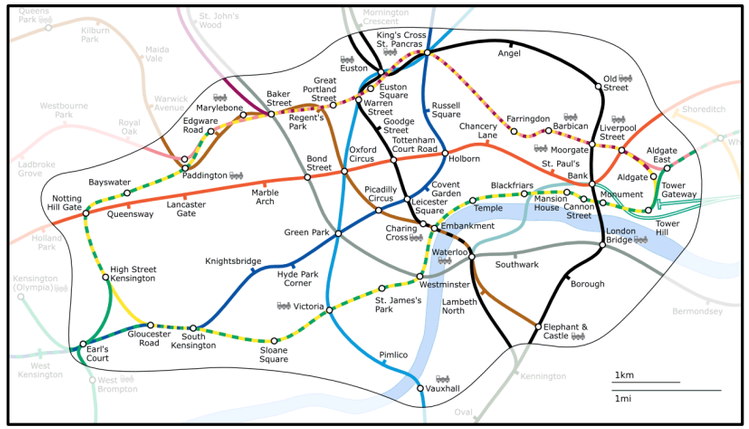




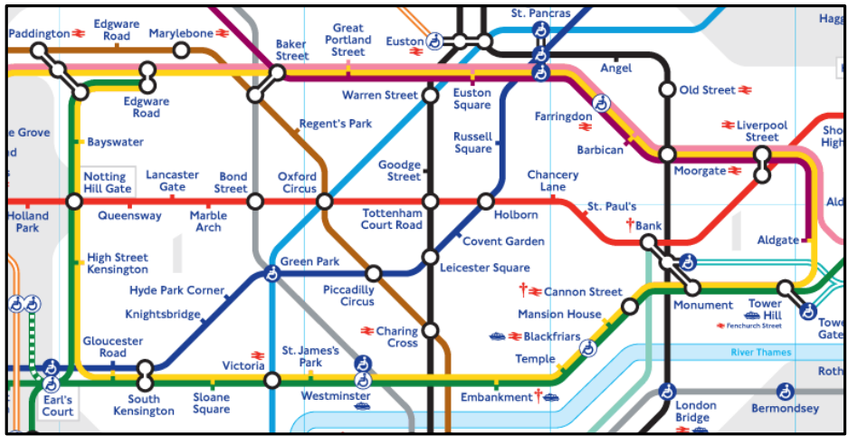


- map/diagram that shows stations connected by metro lines
- focus on topology rather than geography
- goal: easy-to-use visual navigation aid for passengers
  - "*How do I quickly get from A to B?*"
  - "*Where do I need to change trains?*"

What is a Schematic Metro Map?

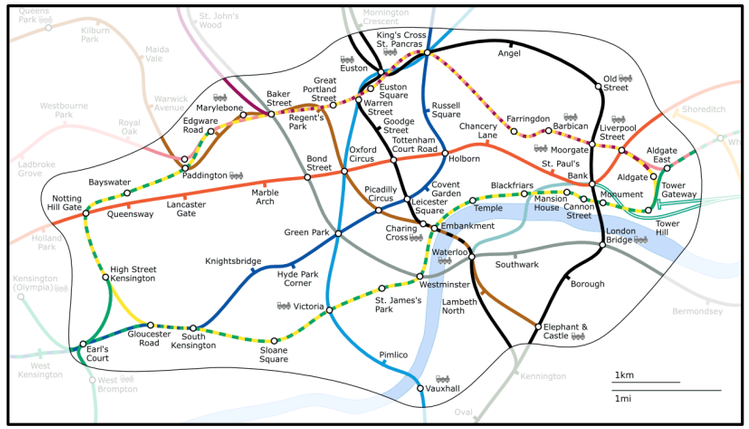




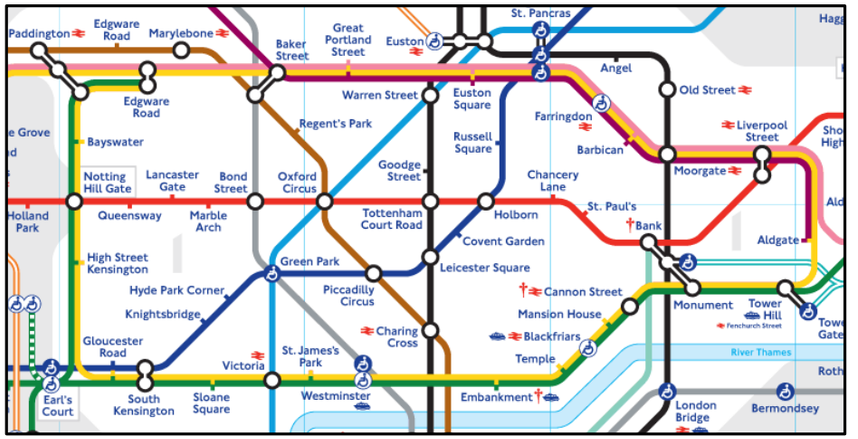


- map/diagram that shows stations connected by metro lines
- focus on topology rather than geography
- goal: easy-to-use visual navigation aid for passengers
  - "*How do I quickly get from A to B?*"
  - "*Where do I need to change trains?*"
- distorts scale and geometry

What is a Schematic Metro Map?

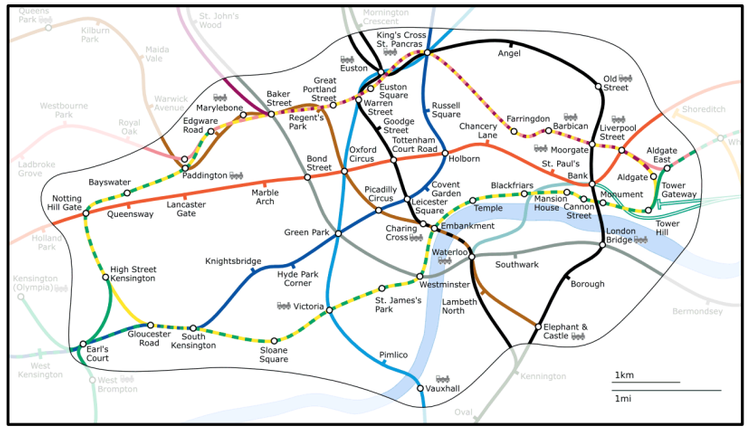


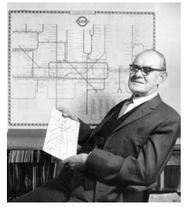


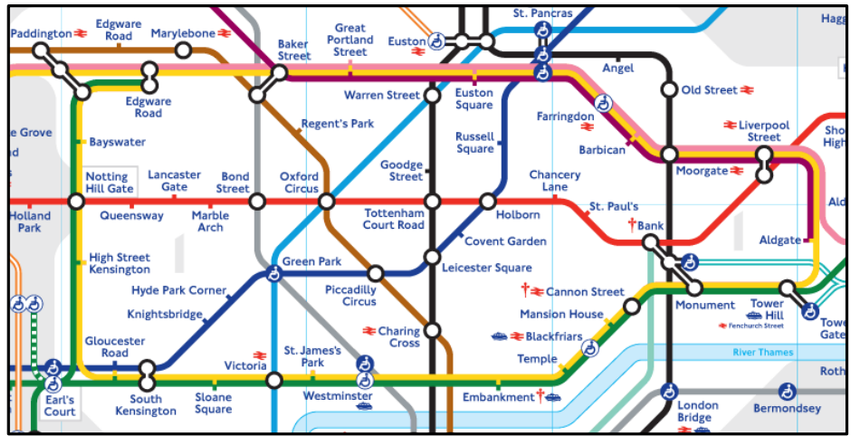


- map/diagram that shows stations connected by metro lines
- focus on topology rather than geography
- goal: easy-to-use visual navigation aid for passengers
  - "*How do I quickly get from A to B?*"
  - "*Where do I need to change trains?*"
- distorts scale and geometry
- metro map design still a largely manual process

What is a Schematic Metro Map?

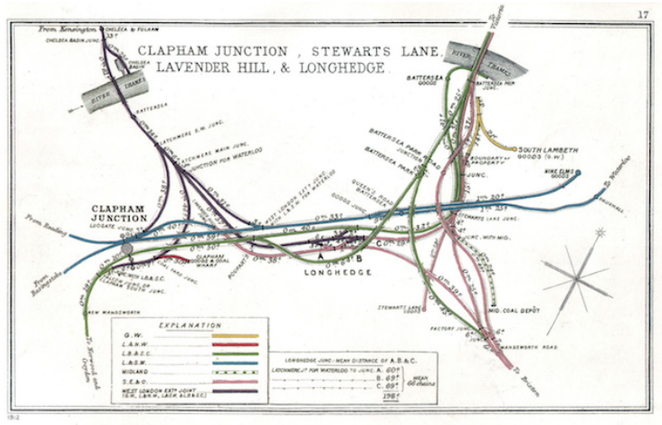






- map/diagram that shows stations connected by metro lines
- focus on topology rather than geography
- goal: easy-to-use visual navigation aid for passengers
  - "*How do I quickly get from A to B?*"
  - "*Where do I need to change trains?*"
- distorts scale and geometry
- metro map design still a largely manual process
- optimizing network layout computationally challenging

Subtasks in Metro Map Layout



+



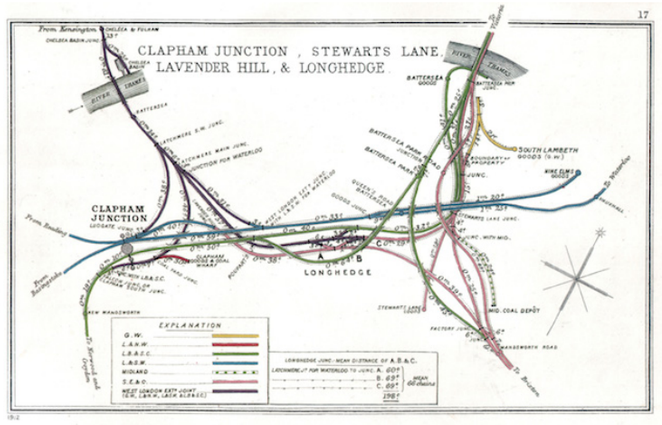
→



Input:

- geographically embedded railway network G
- set $\mathcal{L}$ of metro lines serving G

Subtasks in Metro Map Layout



+



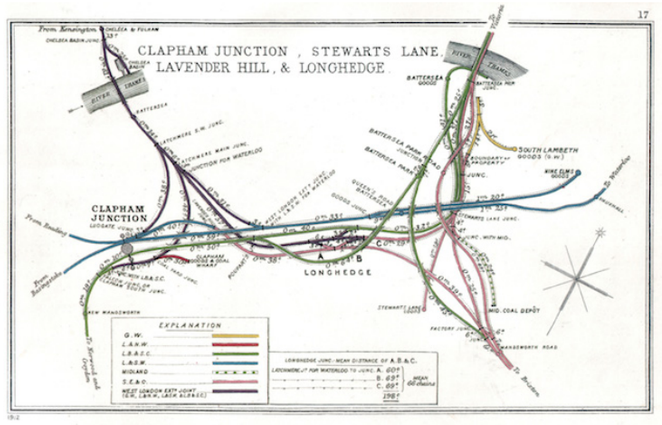
→



Input:
- geographically embedded railway network G
- set $\mathcal{L}$ of metro lines serving G

Output:
- **optimal** metro map layout (whatever it means)

Subtasks in Metro Map Layout



+



→

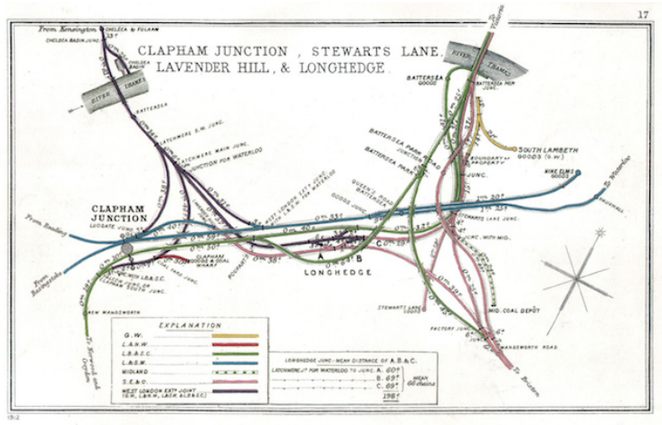


Input:
- geographically embedded railway network G
- set $\mathcal{L}$ of metro lines serving G

Output:
- **optimal** metro map layout (whatever it means)

Divide the task into several subtasks:

Subtasks in Metro Map Layout



+



→

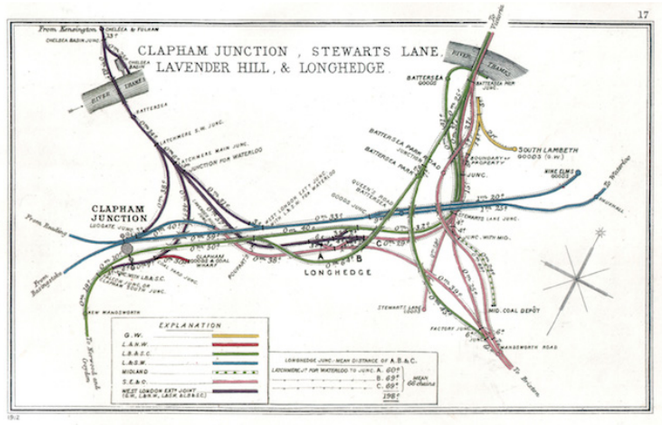


Input:
- geographically embedded railway network G
- set $\mathcal{L}$ of metro lines serving G

Output:
- **optimal** metro map layout (whatever it means)

Divide the task into several subtasks:
- rendering and design choices

Subtasks in Metro Map Layout



+



→

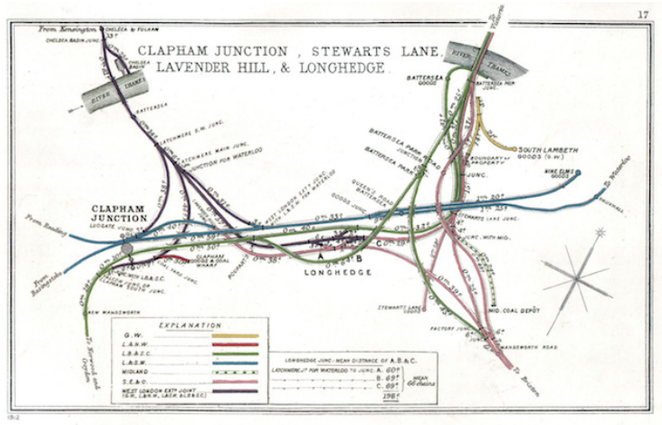


Input:
- geographically embedded railway network G
- set $\mathcal{L}$ of metro lines serving G

Output:
- **optimal** metro map layout (whatever it means)

Divide the task into several subtasks:
- rendering and design choices
- network layout

Subtasks in Metro Map Layout



+



→

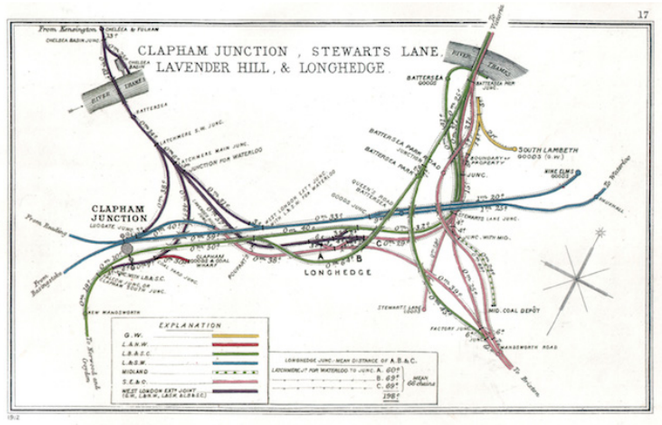


Input:
- geographically embedded railway network G
- set $\mathcal{L}$ of metro lines serving G

Output:
- **optimal** metro map layout (whatever it means)

Divide the task into several subtasks:
- rendering and design choices
- network layout
- station labeling

Subtasks in Metro Map Layout



+



→

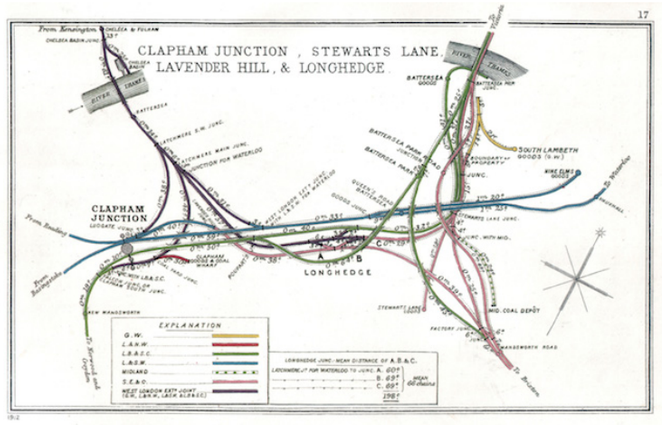


Input:
- geographically embedded railway network G
- set $\mathcal{L}$ of metro lines serving G

Output:
- **optimal** metro map layout (whatever it means)

Divide the task into several subtasks:
- rendering and design choices
- network layout
- station labeling
- metro line routing

Subtasks in Metro Map Layout



+



→



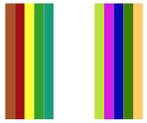
Input:
- geographically embedded railway network G
- set $\mathcal{L}$ of metro lines serving G

Output:
- **optimal** metro map layout (whatever it means)

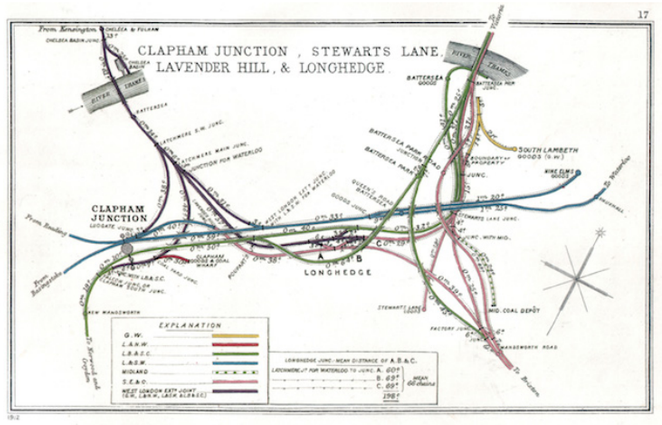
Divide the task into several subtasks:

- rendering and design choices
- network layout
- station labeling
- metro line routing

colors



Subtasks in Metro Map Layout



+



→



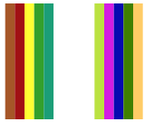
Input:
- geographically embedded railway network G
- set $\mathcal{L}$ of metro lines serving G

Output:
- **optimal** metro map layout (whatever it means)

Divide the task into several subtasks:

- rendering and design choices
- network layout
- station labeling
- metro line routing

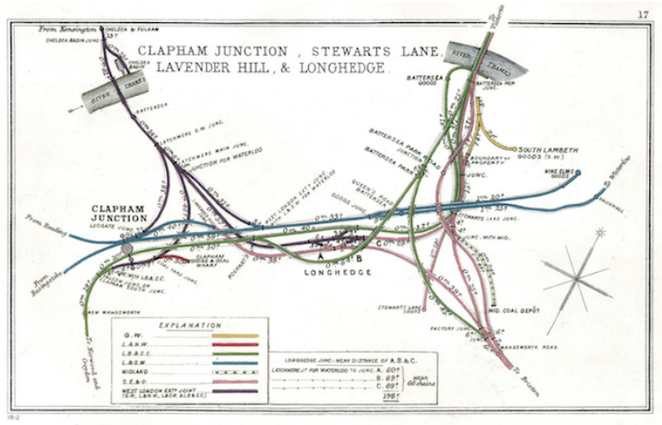
colors



fonts

Paddington
PADDINGTON
Paddington

Subtasks in Metro Map Layout



+



→



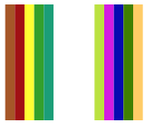
Input:
- geographically embedded railway network G
- set $\mathcal{L}$ of metro lines serving G

Output:
- **optimal** metro map layout (whatever it means)

Divide the task into several subtasks:

- rendering and design choices
- network layout
- station labeling
- metro line routing

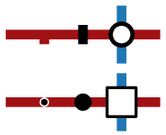
colors



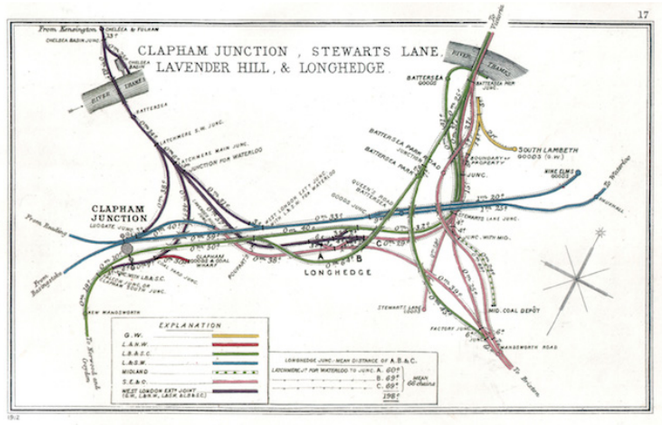
fonts

Paddington
PADDINGTON
Paddington

symbols



Subtasks in Metro Map Layout



+



→



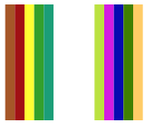
Input:
- geographically embedded railway network G
- set $\mathcal{L}$ of metro lines serving G

Output:
- **optimal** metro map layout (whatever it means)

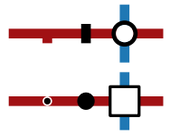
Divide the task into several subtasks:

- rendering and design choices
- network layout
- station labeling
- metro line routing

colors fonts symbols

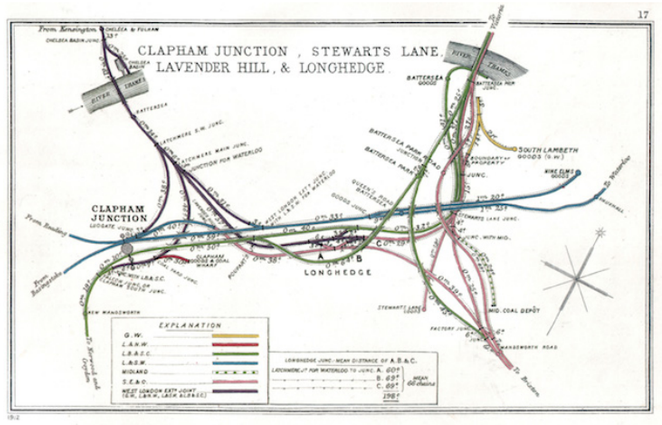


Paddington
PADDINGTON
Paddington



→ very salient,
but not a computational problem

Subtasks in Metro Map Layout



+



→

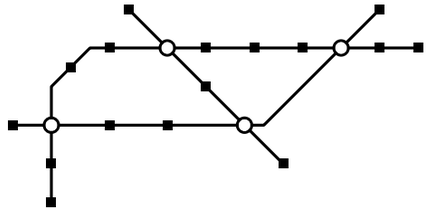


Input:
- geographically embedded railway network G
- set $\mathcal{L}$ of metro lines serving G

Output:
- **optimal** metro map layout (whatever it means)

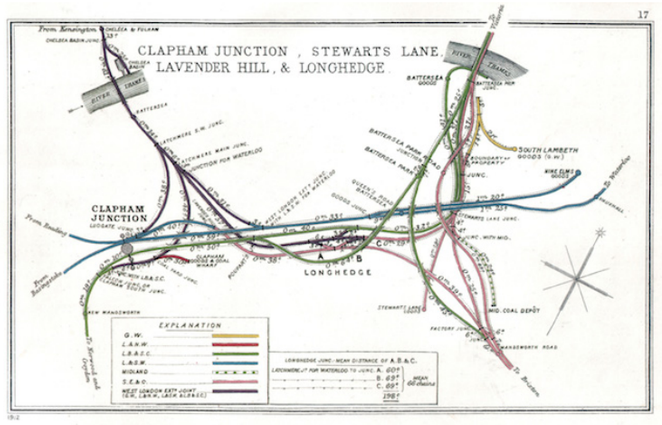
Divide the task into several subtasks:

- rendering and design choices
- network layout
- station labeling
- metro line routing



determine geometry of network layout

Subtasks in Metro Map Layout



+



→

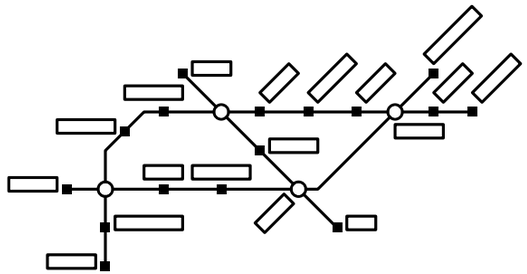


Input:
- geographically embedded railway network G
- set $\mathcal{L}$ of metro lines serving G

Output:
- **optimal** metro map layout (whatever it means)

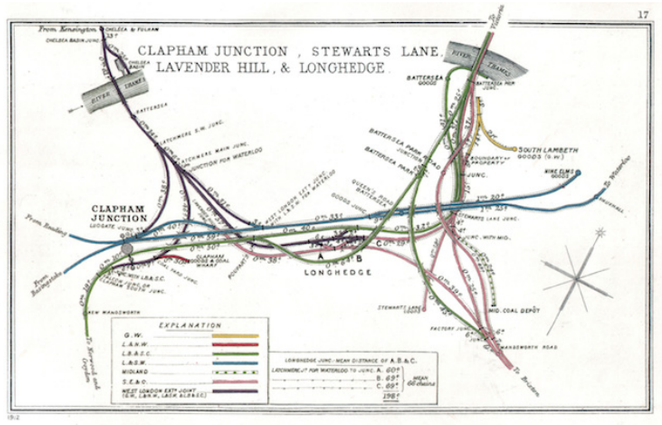
Divide the task into several subtasks:

- rendering and design choices
- network layout
- station labeling
- metro line routing



determine positions of station names

Subtasks in Metro Map Layout



+



→

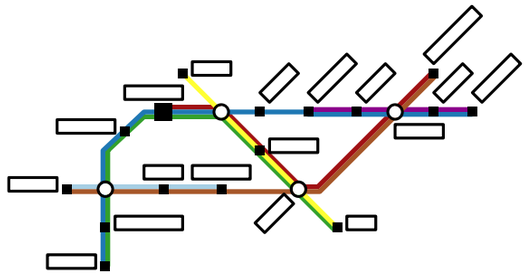


Input:
- geographically embedded railway network G
- set $\mathcal{L}$ of metro lines serving G

Output:
- **optimal** metro map layout (whatever it means)

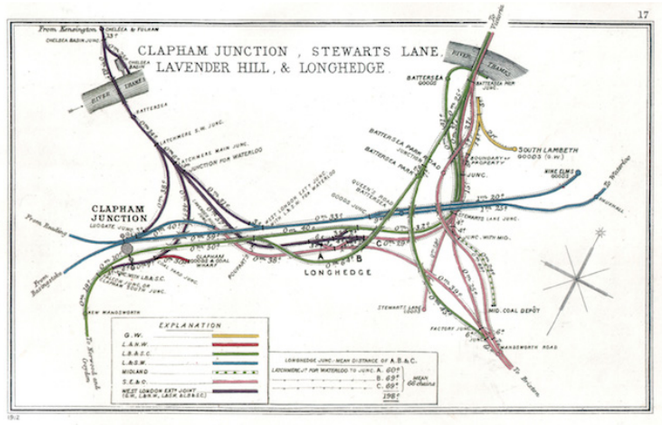
Divide the task into several subtasks:

- rendering and design choices
- network layout
- station labeling
- metro line routing



determine line routing and ordering of bundles

Subtasks in Metro Map Layout



+



→



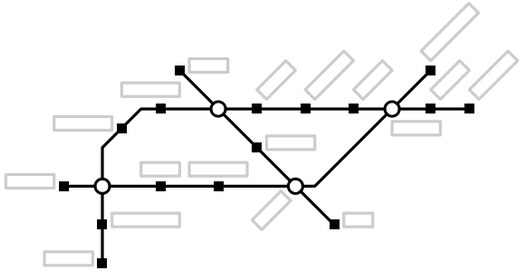
Input:
- geographically embedded railway network G
- set $\mathcal{L}$ of metro lines serving G

Output:
- **optimal** metro map layout (whatever it means)

Divide the task into several subtasks:

- rendering and design choices
- network layout
- station labeling
- metro line routing

focus today



Formalizing the Network Layout Problem

Given

- graph $G = (V, E)$ geometrically embedded in $\mathbb{R}^2$
- vertex set V (stations)
- edge set E (rail links)
- set of paths $\mathcal{L}$ (metro lines in G)

Find a **schematic** layout of $(G, \mathcal{L})$ that

- satisfies a set of layout constraints
- optimizes a set of quality criteria

Formalizing the Network Layout Problem

Given
- graph $G = (V, E)$ geometrically embedded in $\mathbb{R}^2$
- vertex set V (stations)
- edge set E (rail links)
- set of paths $\mathcal{L}$ (metro lines in G)

Find a **schematic** layout of $(G, \mathcal{L})$ that
- satisfies a set of layout constraints
- optimizes a set of quality criteria

But what are the constraints and quality criteria?

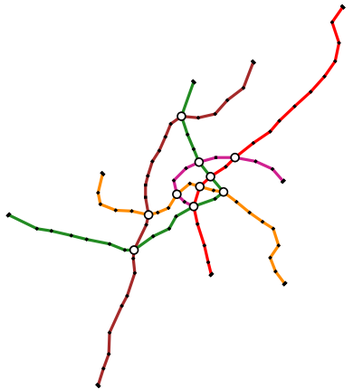
$\rightarrow$ extract common principles of existing, manually designed metro maps

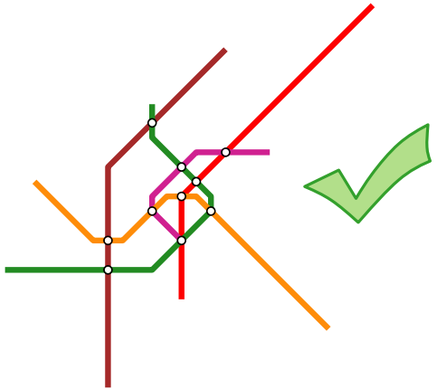


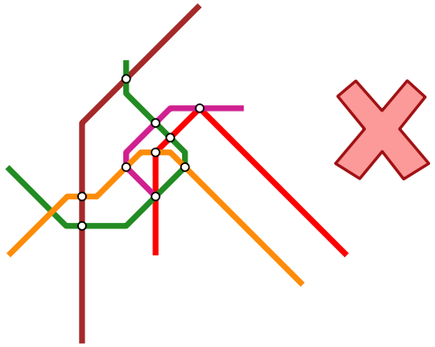
Design Rules

(R1) Do not change the network topology.

- no new crossings
- no changes in circular vertex orders





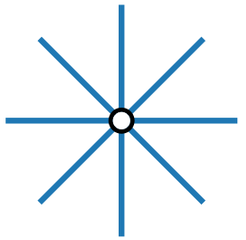


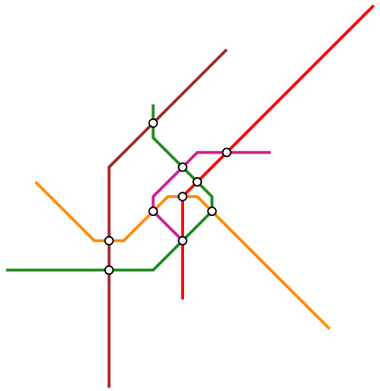
Design Rules

(R1) Do not change the network topology.

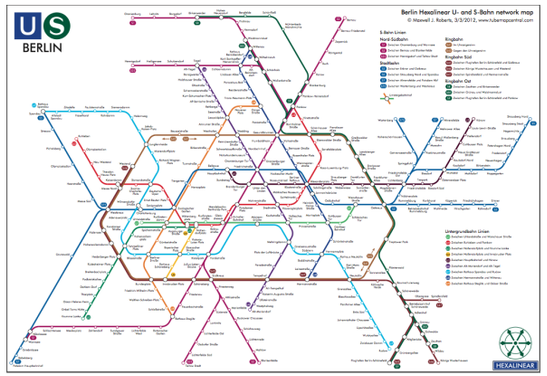
(R2) Restrict edge orientations.

- mostly octilinear (octolinear) orientation systems
- also curvilinear and other alternative orientation systems









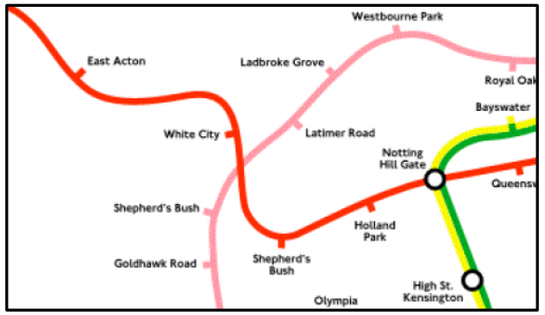
Design Rules

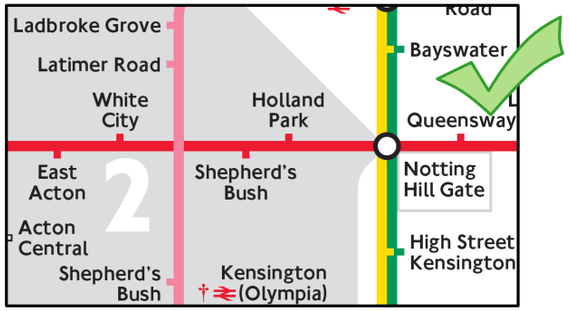
(R1) Do not change the network topology.

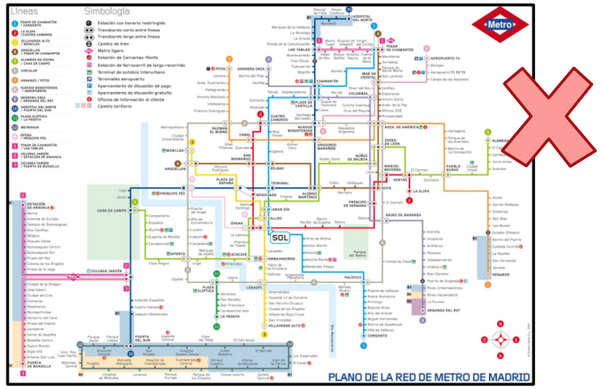
(R2) Restrict edge orientations.

(R3) Draw metro lines as simple and monotone as possible.

- avoid bends
- prefer obtuse bend angles
- for curves: prefer uniform curvature, few inflection points







Design Rules

(R1) Do not change the network topology.

(R2) Restrict edge orientations.

(R3) Draw metro lines as simple and monotone as possible.

(R4) Let lines pass straight through interchanges.

- avoids visual ambiguities in complex stations



Design Rules

(R1) Do not change the network topology.

(R2) Restrict edge orientations.

(R3) Draw metro lines as simple and monotone as possible.

(R4) Let lines pass straight through interchanges.

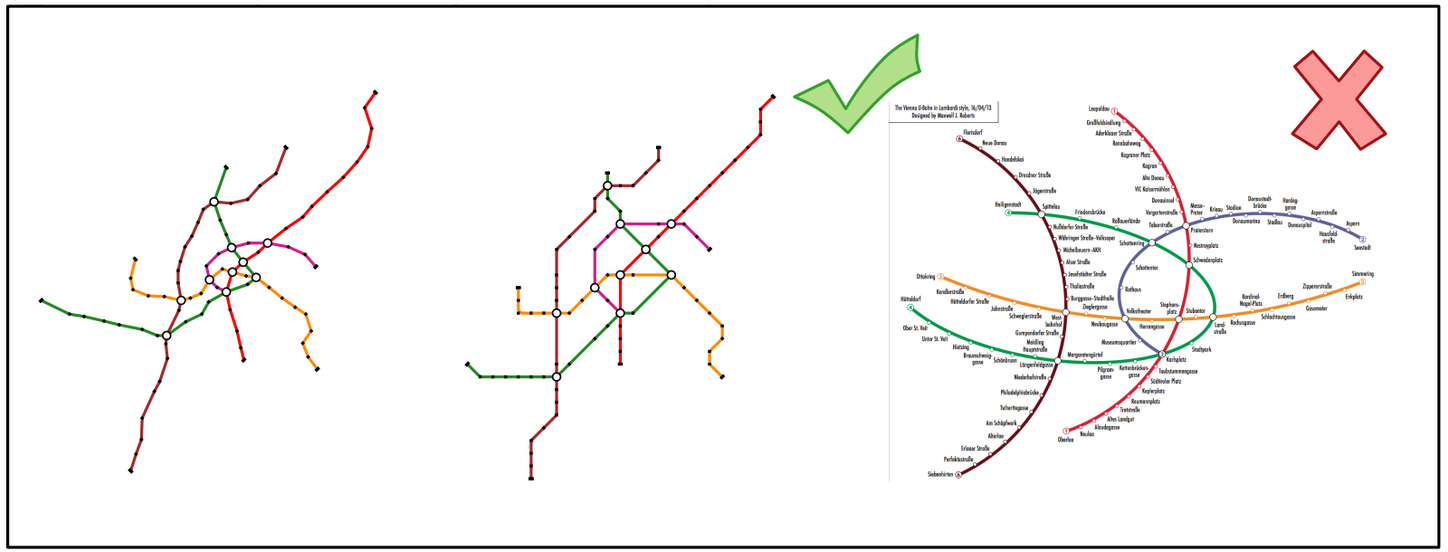
(R5) Use large angular resolution in stations.

- distributes edges evenly for balanced appearance



Design Rules

(R1)
(R2)
(R3)
(R4)
(R5)



(R6) Minimize geometric distortion and displacement.

- maintains resemblance to geography
- preserves user's mental map
- applicable locally or globally

topographicity

Design Rules

(R1) Do not change the network topology.

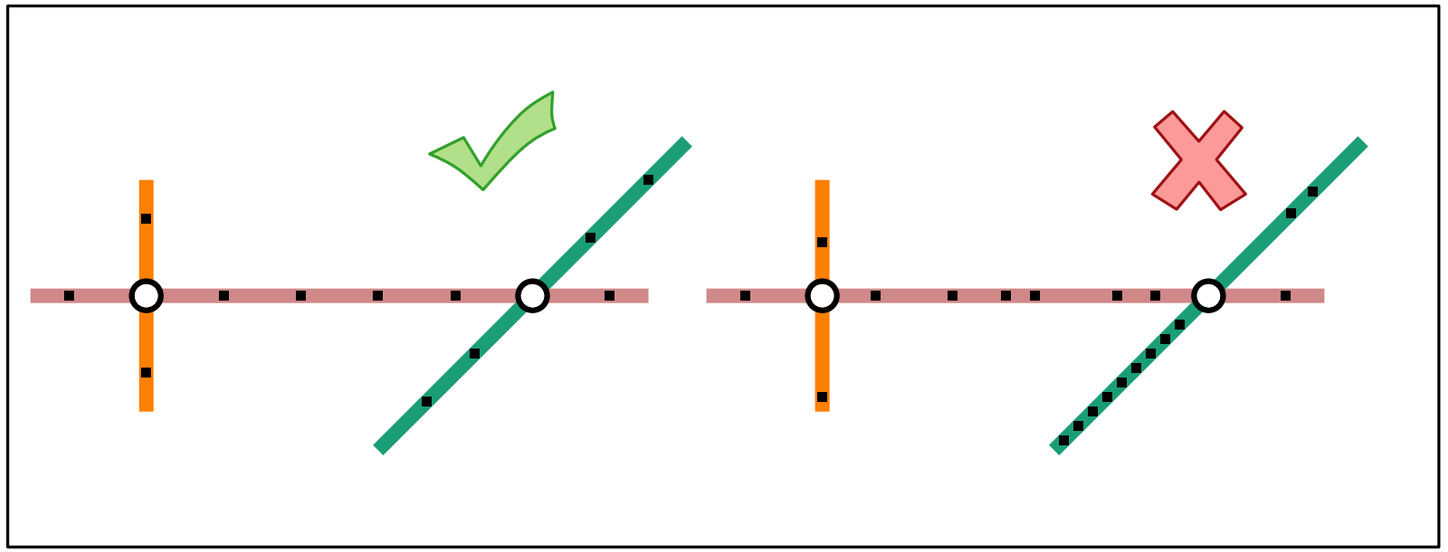
(R2)

(R3)

(R4)

(R5)

(R6)



(R7) Use uniform edge lengths.

- geographic distances less important
- network hop-distances more important
- balanced appearance

Design Rules

(R1) Do not change the network topology.

(R2) Restrict edge orientations.

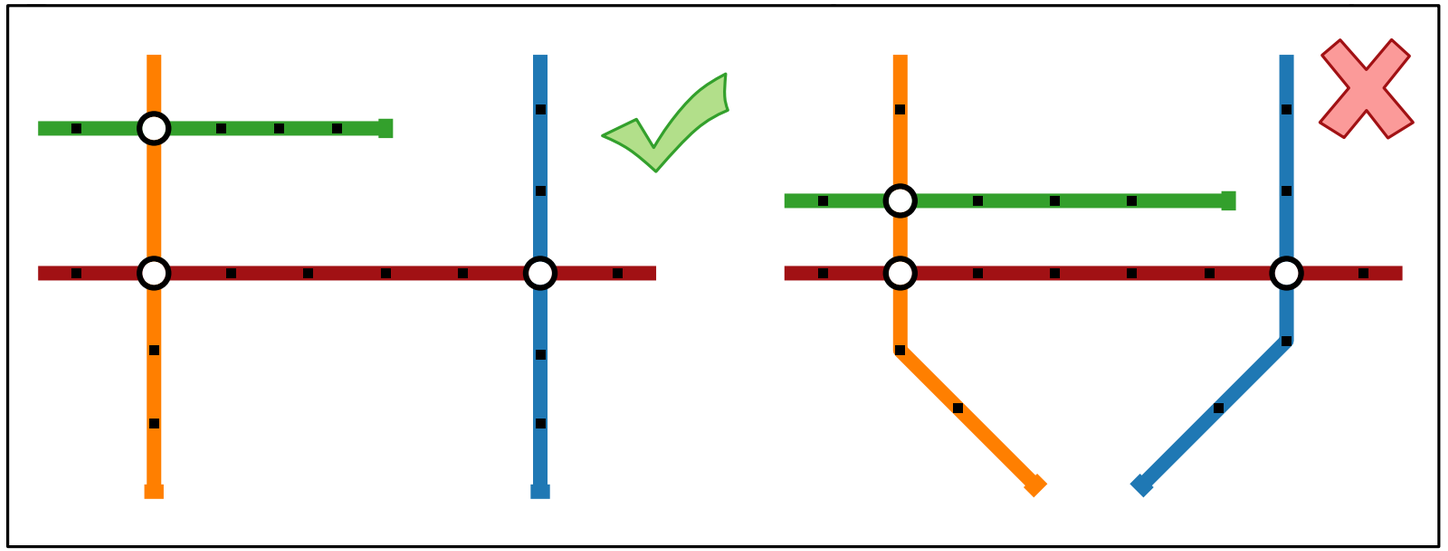
(R3)

(R4)

(R5)

(R6)

(R7)



(R8) Keep unrelated features apart.

- guarantees minimum clearance between features
- avoids ambiguities

Design Rules

(R1) Do not change the network topology.

(R2) Restrict edge orientations.

(R3)

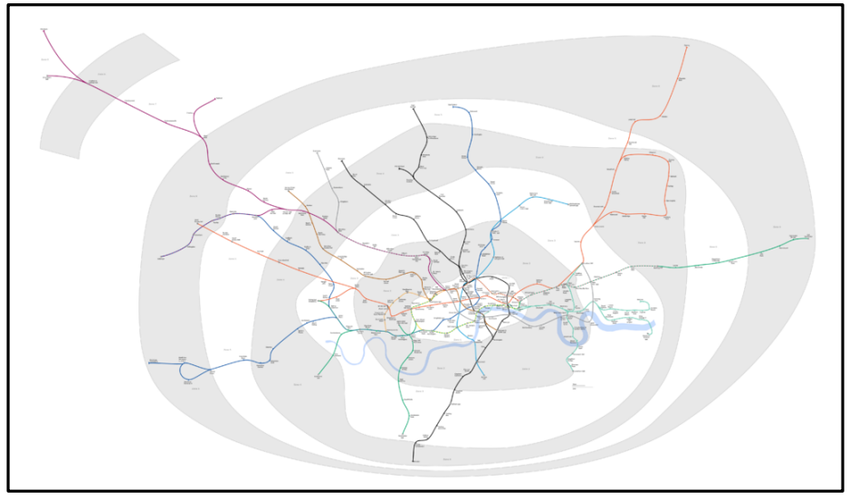
(R4)

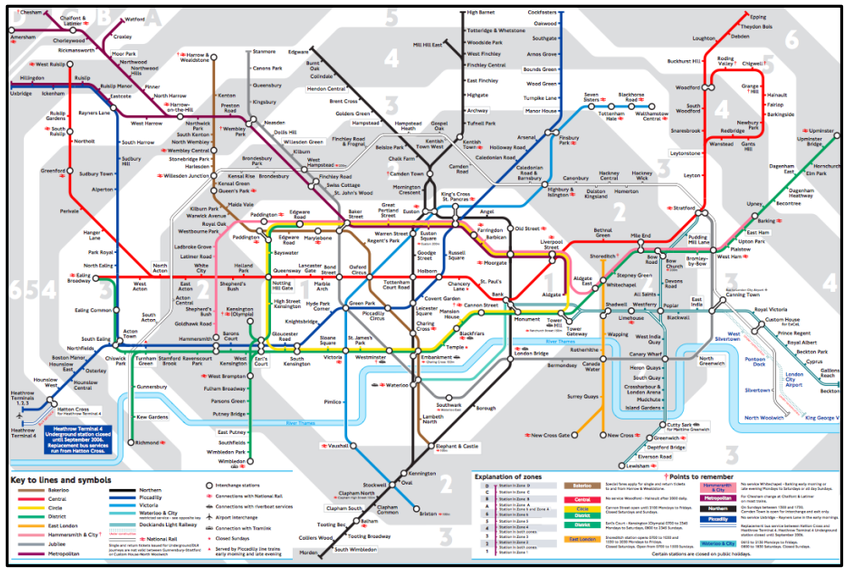
(R5)

(R6)

(R7)

(R8)





(R9) Avoid large empty spaces.

- balances local feature density
- possibly fill gaps with legends

Design Rules

(R1) Do not change the network topology.

(R2) Restrict edge orientations.

(R3) Draw metro lines as simple and monotone as possible.

(R4) Let lines pass straight through interchanges.

(R5) Use large angular resolution in stations.

(R6) Minimize geometric distortion and displacement.

(R7) Use uniform edge lengths.

(R8) Keep unrelated features apart.

(R9) Avoid large empty spaces.

Design Rules

(R1) Do not change the network topology.

(R2) Restrict edge orientations.

(R3) Draw metro lines as simple and monotone as possible.

(R4) Let lines pass straight through interchanges.

(R5) Use large angular resolution in stations.

(R6) Minimize geometric distortion and displacement.

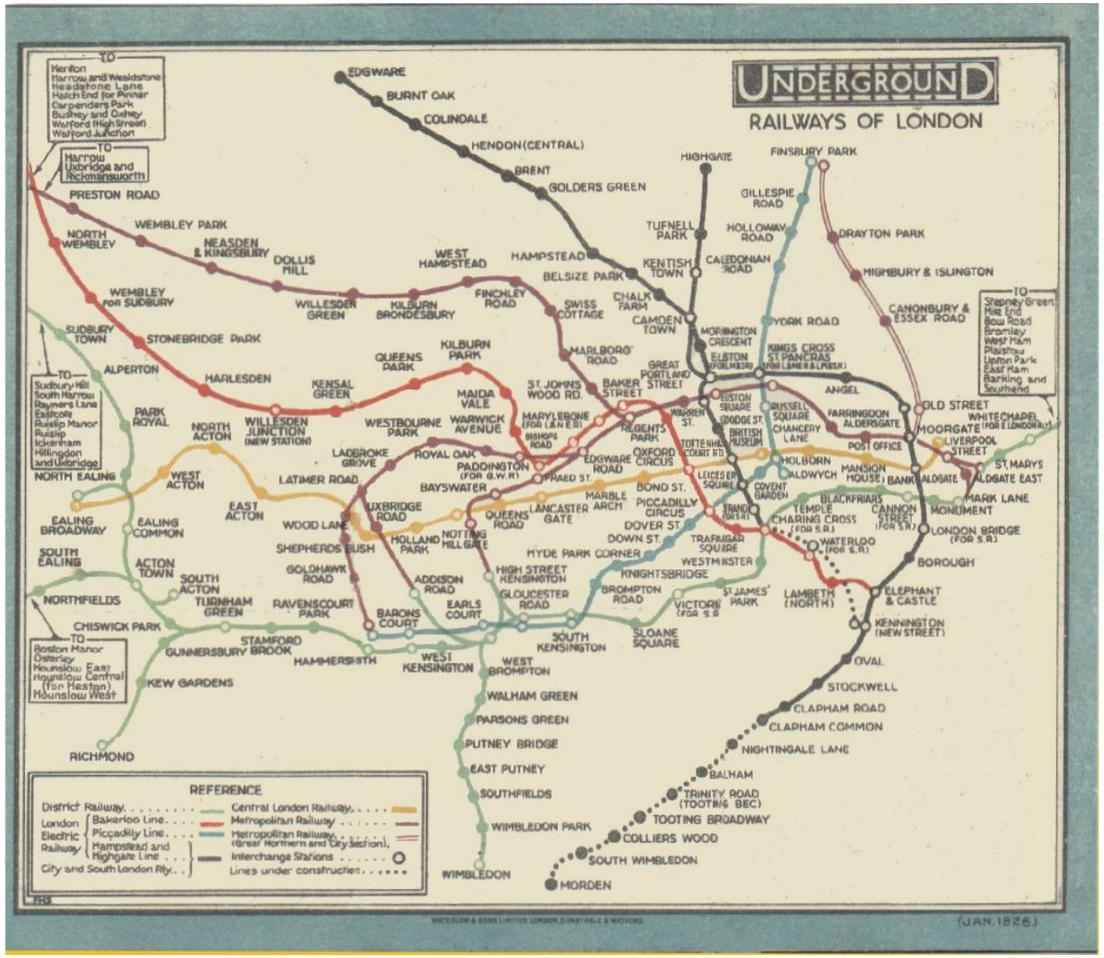
(R7) Use uniform edge lengths.

(R8) Keep unrelated features apart.

(R9) Avoid large empty spaces.

$\rightarrow$ **rules are potentially conflicting and need priorities**

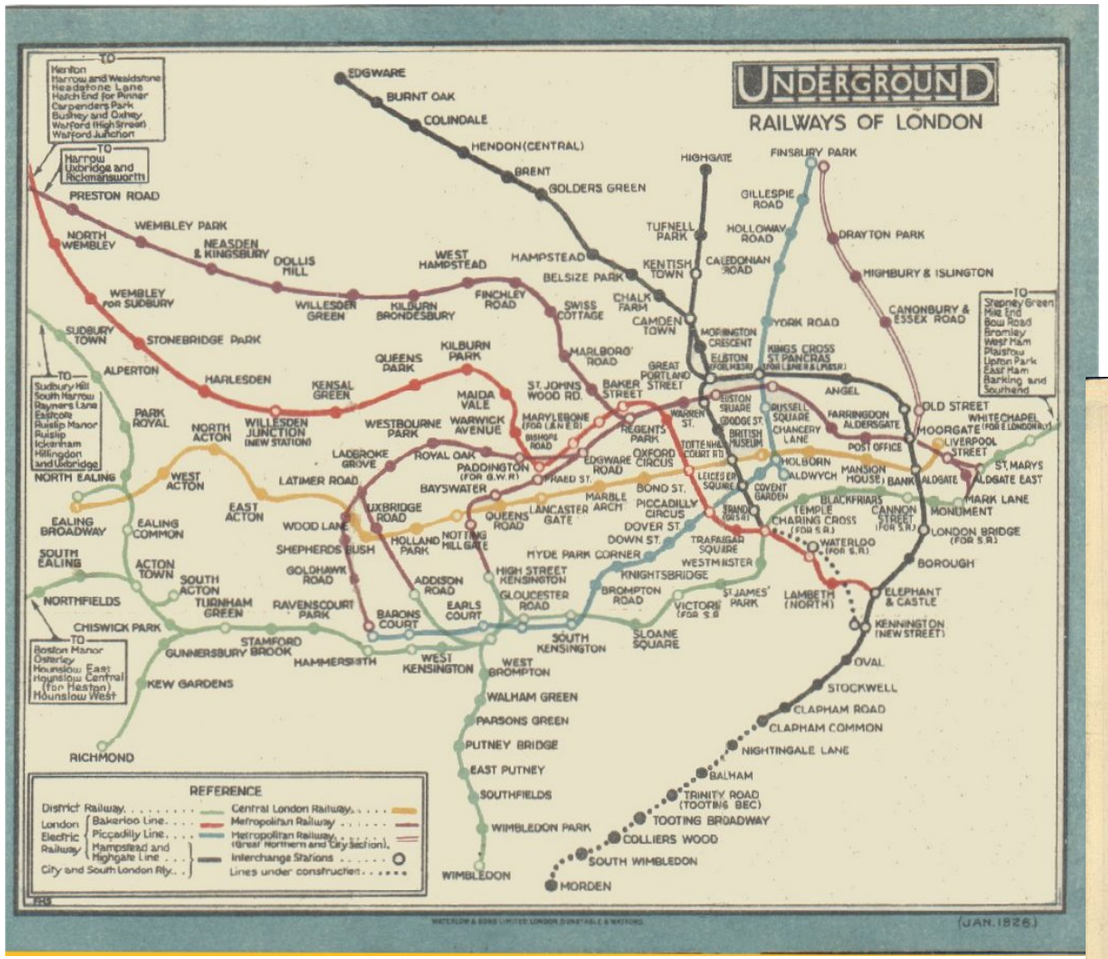
A Bit of History



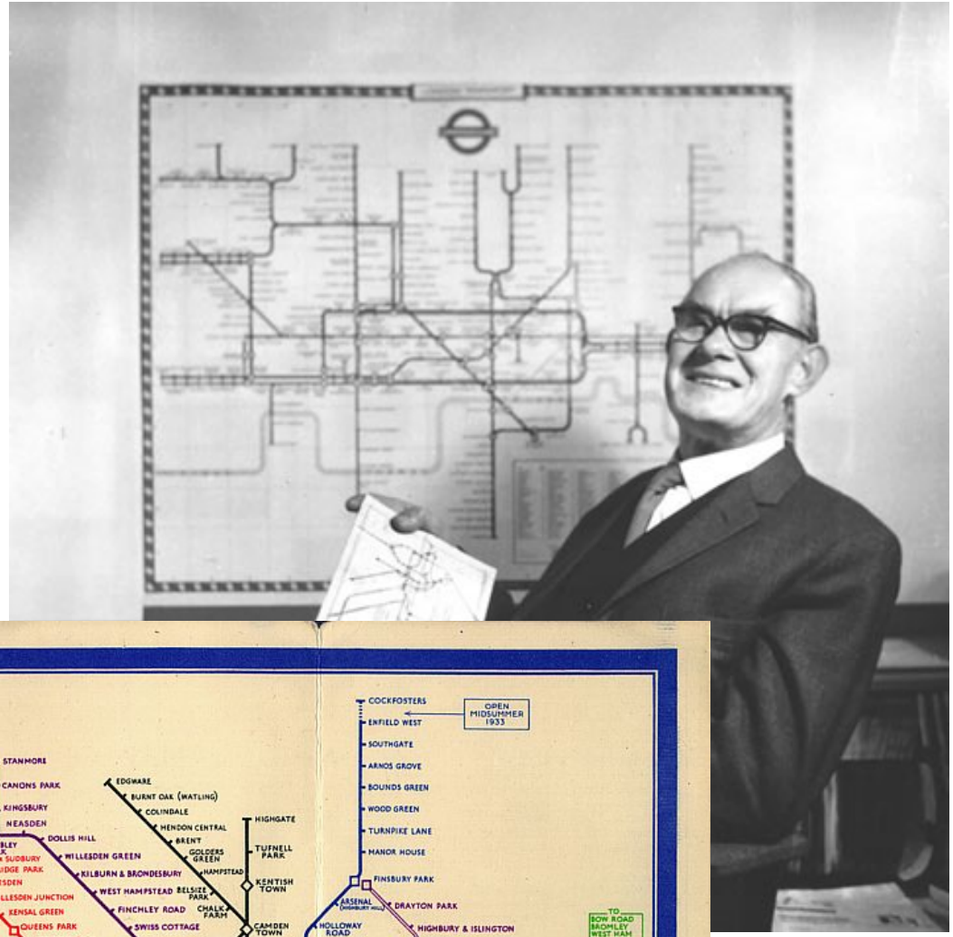
London 1927 (Fred H. Stingemore)

(c) Mike Yashworth

A Bit of History



London 1927 (Fred H. Stingemore)

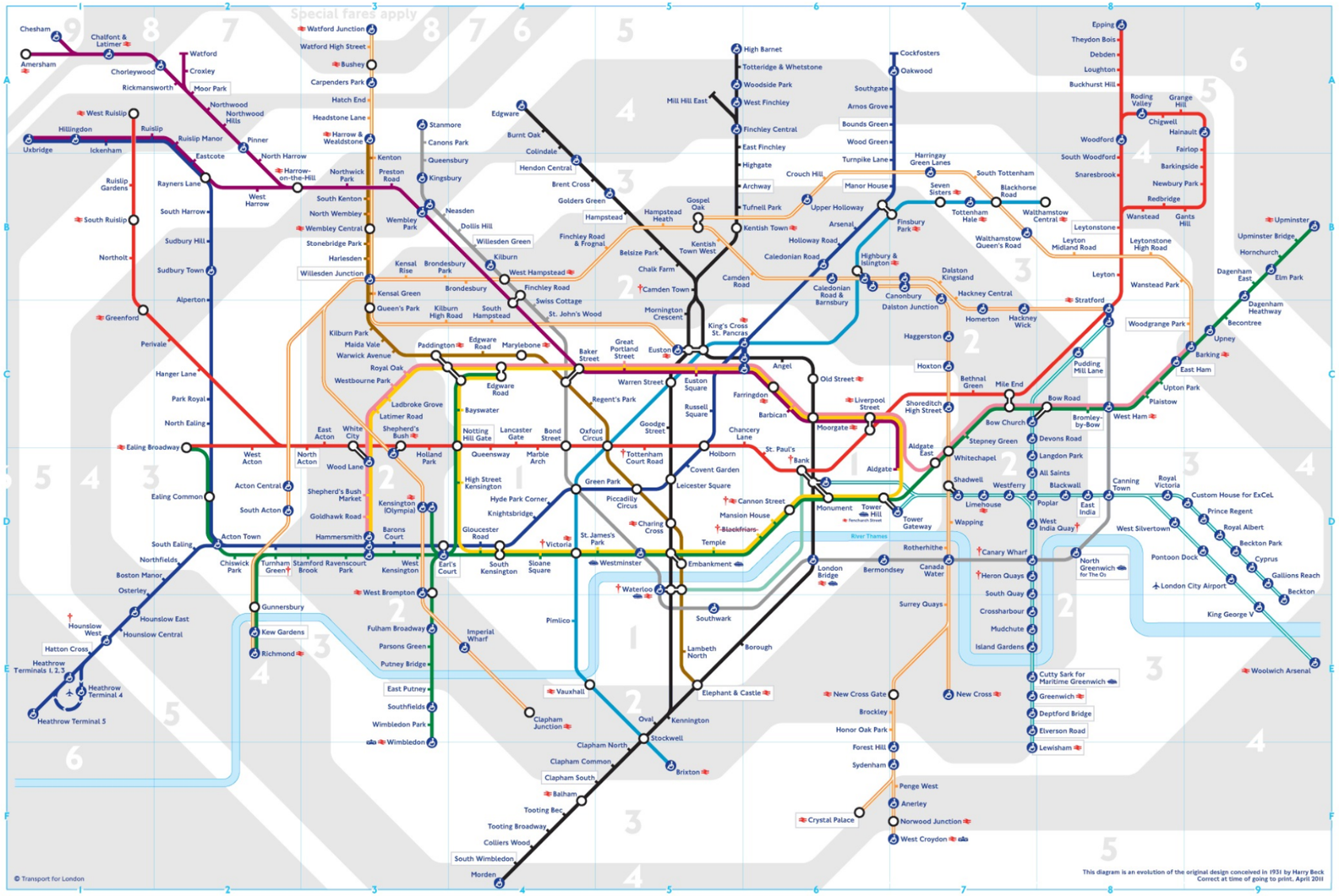


Henry Beck
London 1933



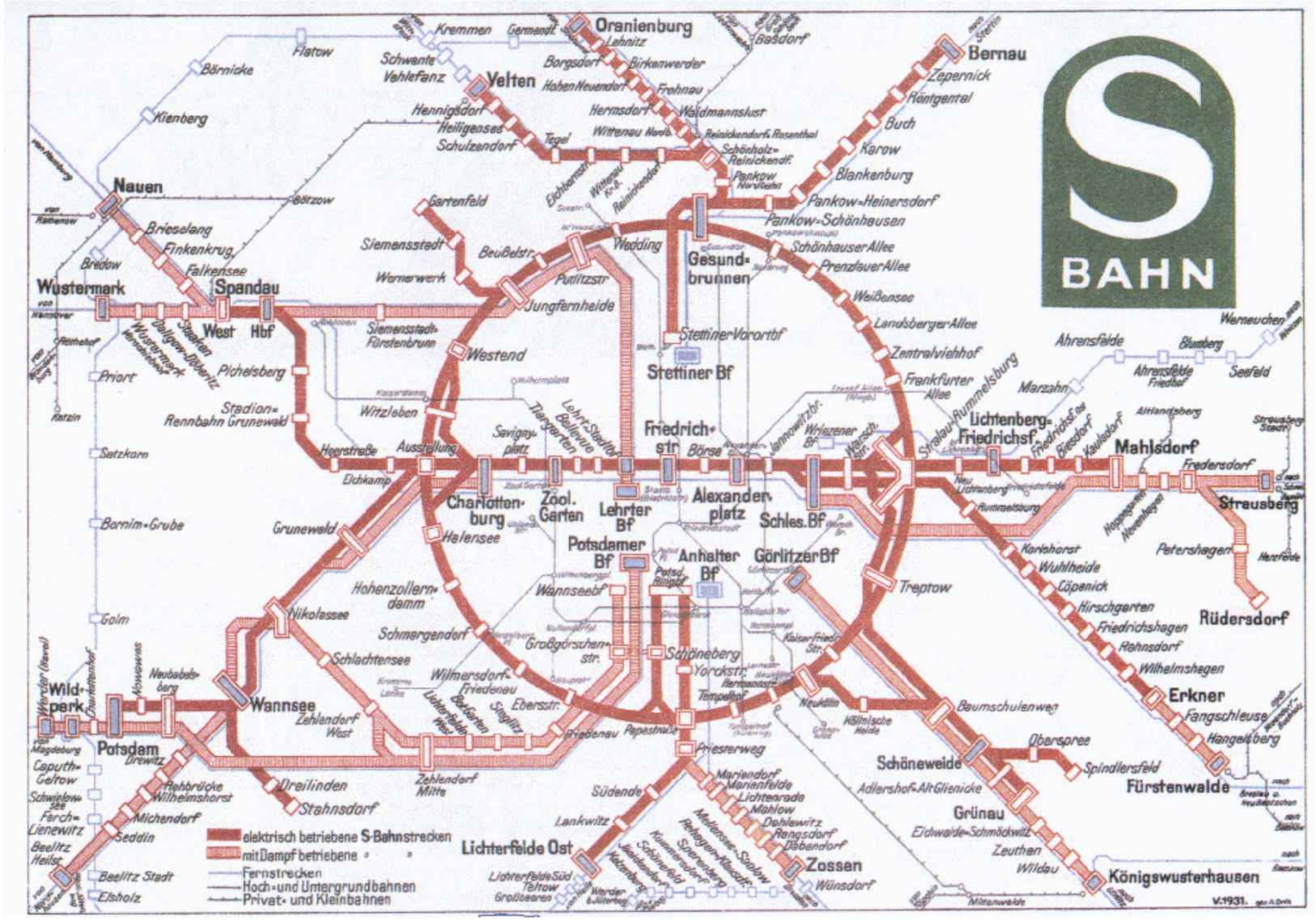
(c) Mike Yashworth

A Bit of History



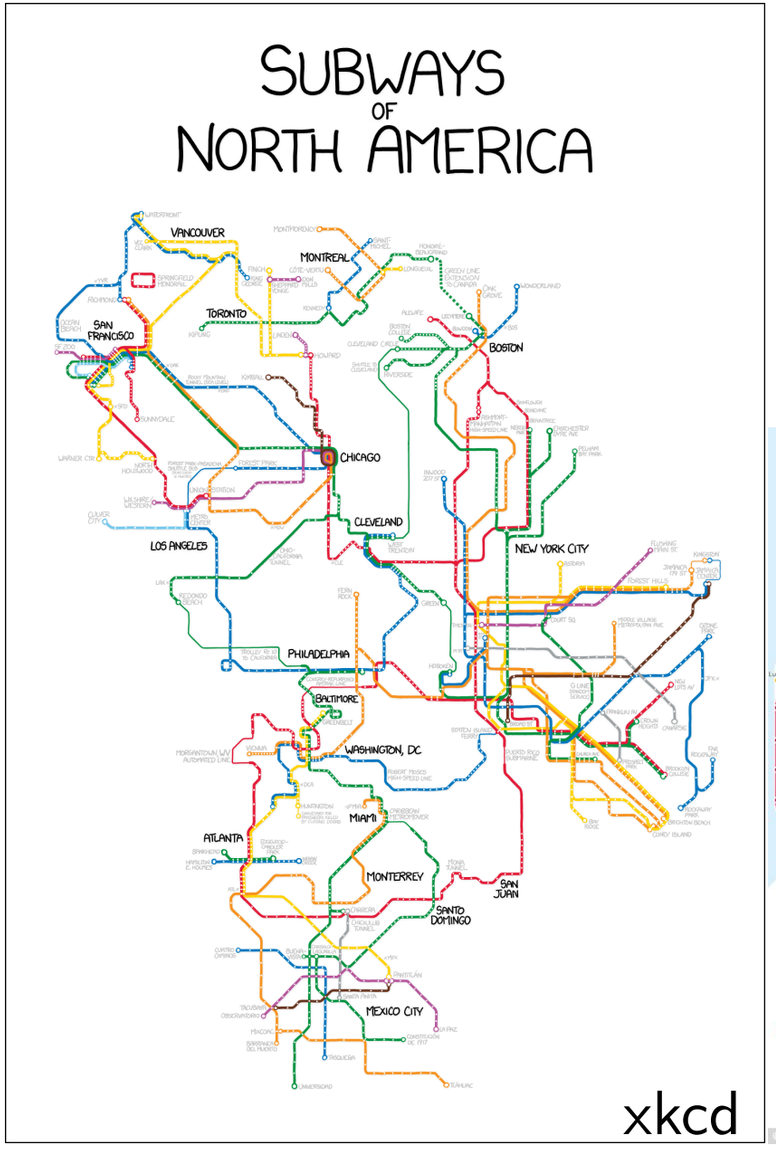
Tube Map
voted Design Icon 2006
(2nd after Concorde)

A Bit of History

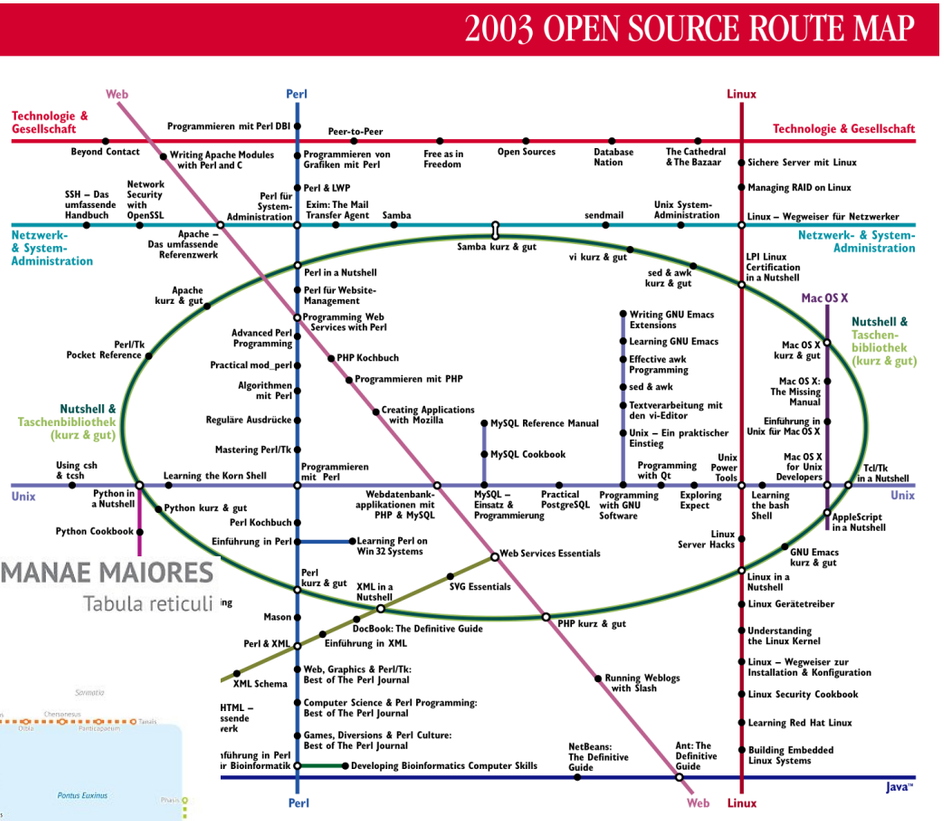


Berlin 1931 (redrawn by Maxwell Roberts)

A Bit of History



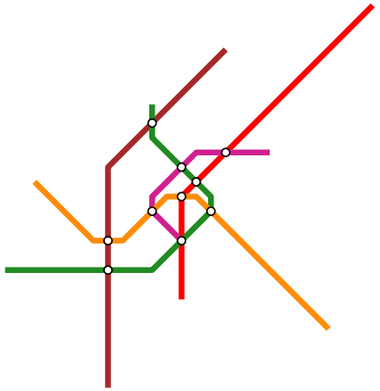




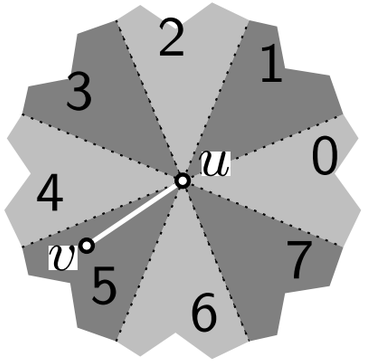


Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

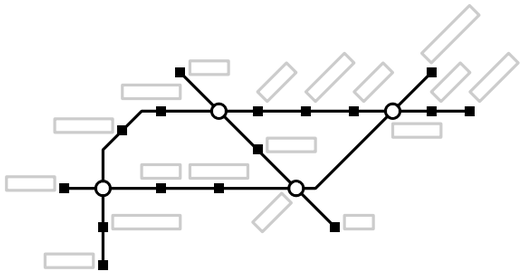
Visualization of Graphs



Lecture 12:
Octilinear Graph Drawing
Metro Map Layout



Part II:
Complexity and Path-Based Schematization



Alexander Wolff

Complexity

Theorem 1. [Nöllenburg 2005]
For an embedded graph G (vertex degrees ≤ 8) **bend minimization** (R3) is NP-hard if **preserving topology** (R1) and **octilinearity** (R2) are required.

Complexity

Theorem 1. [Nöllenburg 2005]
For an embedded graph G (vertex degrees ≤ 8) **bend minimization** (R3) is NP-hard if **preserving topology** (R1) and **octilinearity** (R2) are required.

Sketch of proof.

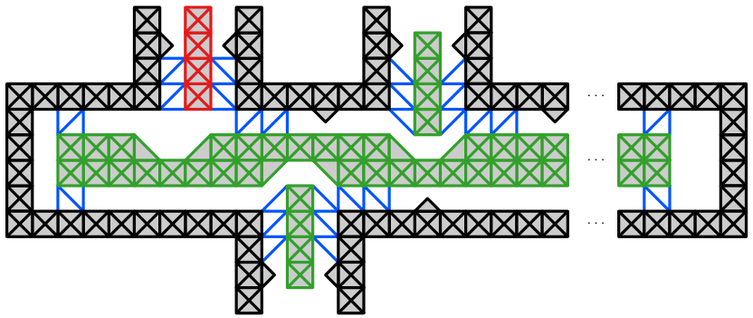
Reduction from Boolean satisfiability problem Planar-3Sat using rigid "mechanical" gadgets

Complexity

Theorem 1. [Nöllenburg 2005]
For an embedded graph G (vertex degrees ≤ 8) **bend minimization** (R3) is NP-hard if **preserving topology** (R1) and **octilinearity** (R2) are required.

Sketch of proof.

Reduction from Boolean satisfiability problem Planar-3Sat using rigid "mechanical" gadgets

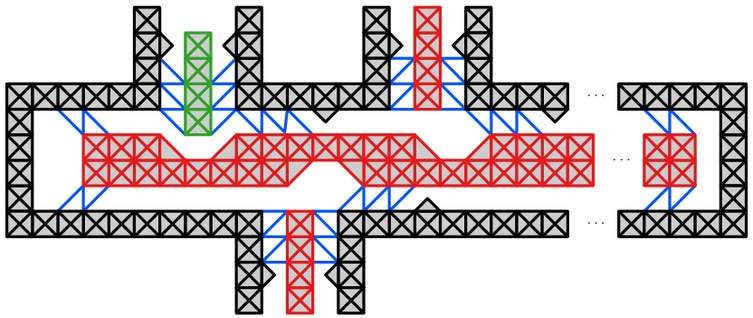


Complexity

Theorem 1. [Nöllenburg 2005]
For an embedded graph G (vertex degrees ≤ 8) **bend minimization** (R3) is NP-hard if **preserving topology** (R1) and **octilinearity** (R2) are required.

Sketch of proof.

Reduction from Boolean satisfiability problem Planar-3Sat using rigid "mechanical" gadgets

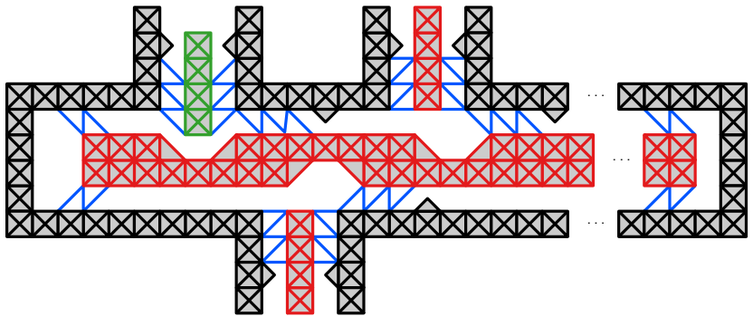


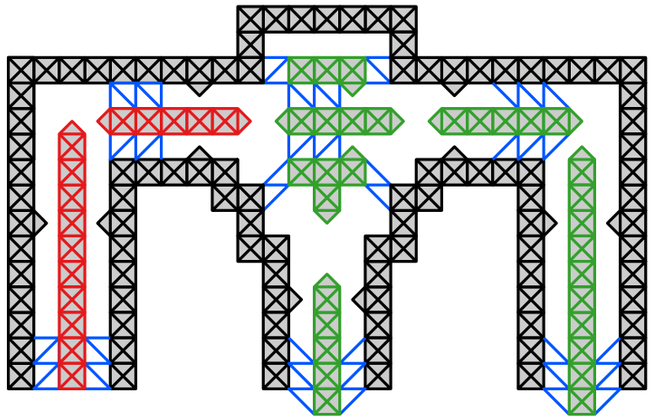
Complexity

Theorem 1. [Nöllenburg 2005]
For an embedded graph G (vertex degrees ≤ 8) **bend minimization** (R3) is NP-hard if **preserving topology** (R1) and **octilinearity** (R2) are required.

Sketch of proof.

Reduction from Boolean satisfiability problem Planar-3Sat using rigid "mechanical" gadgets



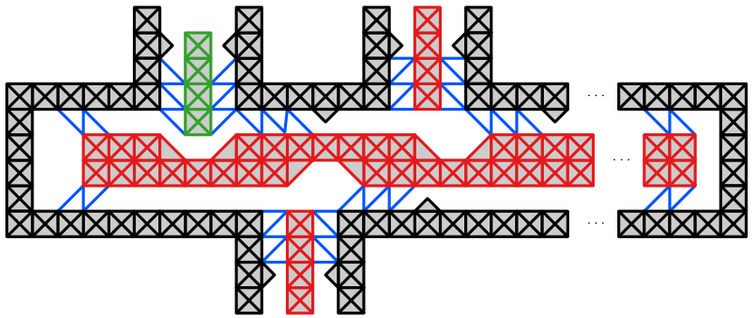


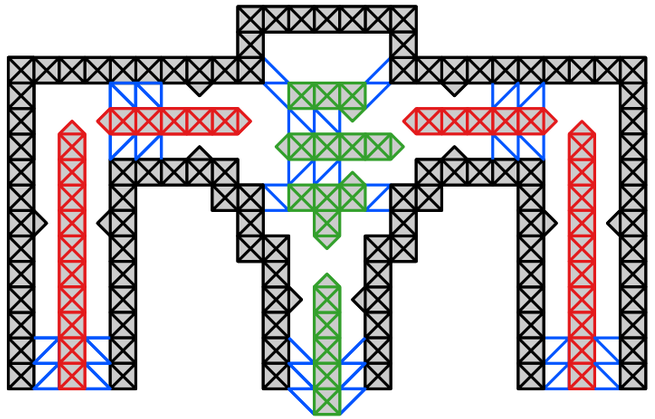
Complexity

Theorem 1. [Nöllenburg 2005]
For an embedded graph G (vertex degrees ≤ 8) **bend minimization** (R3) is NP-hard if **preserving topology** (R1) and **octilinearity** (R2) are required.

Sketch of proof.

Reduction from Boolean satisfiability problem Planar-3Sat using rigid "mechanical" gadgets



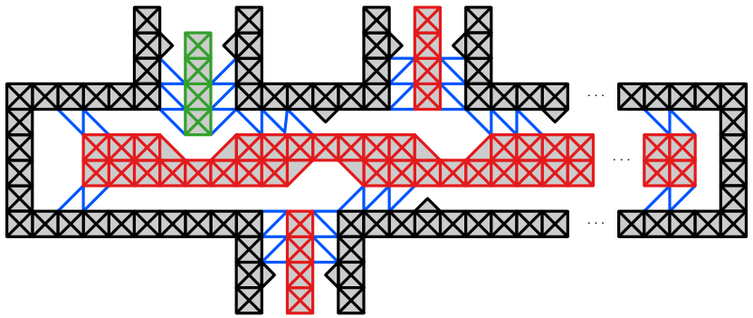


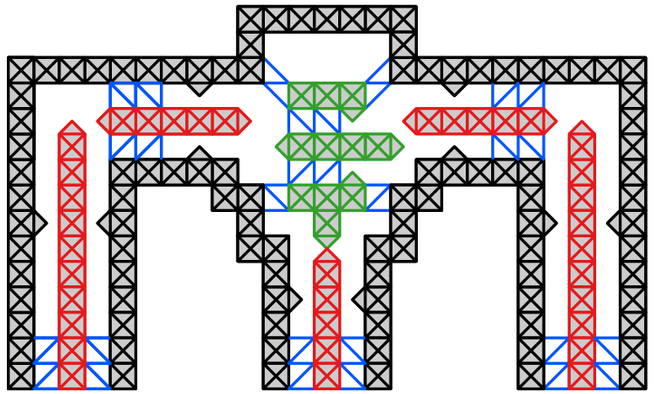
Complexity

Theorem 1. [Nöllenburg 2005]
For an embedded graph G (vertex degrees ≤ 8) **bend minimization** (R3) is NP-hard if **preserving topology** (R1) and **octilinearity** (R2) are required.

Sketch of proof.

Reduction from Boolean satisfiability problem Planar-3Sat using rigid "mechanical" gadgets



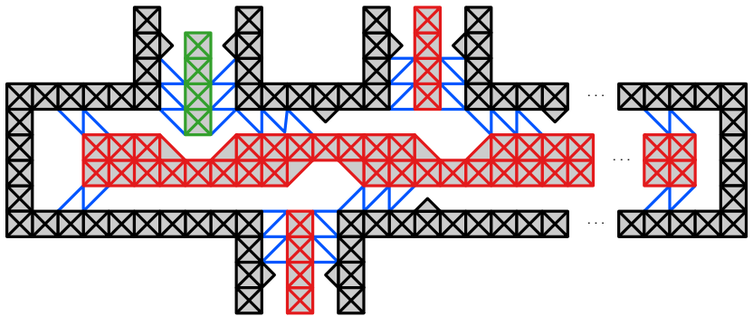


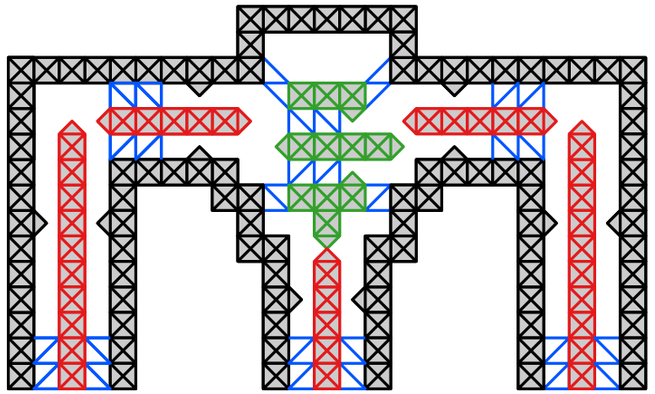
Complexity

Theorem 1. [Nöllenburg 2005]
For an embedded graph G (vertex degrees ≤ 8) **bend minimization** (R3) is NP-hard if **preserving topology** (R1) and **octilinearity** (R2) are required.

Sketch of proof.

Reduction from Boolean satisfiability problem Planar-3Sat using rigid "mechanical" gadgets





Remark.

- no efficient exact algorithms to expect
- same problem without diagonals (rectilinear) is efficiently solvable [Tamassia '87]

Path-Based Schematization

Goal. Solve restricted problem, where G is a path (or polyline)

Path-Based Schematization

Goal. Solve restricted problem, where G is a path (or polyline)

Constraints.
- $\mathcal{C}$-oriented edges (e.g. octilinear) **(R2)**
- bounded displacement **(R6)**

Path-Based Schematization

Goal. Solve restricted problem, where G is a path (or polyline)

Constraints.
- $\mathcal{C}$-oriented edges (e.g. octilinear) **(R2)**
- bounded displacement **(R6)**

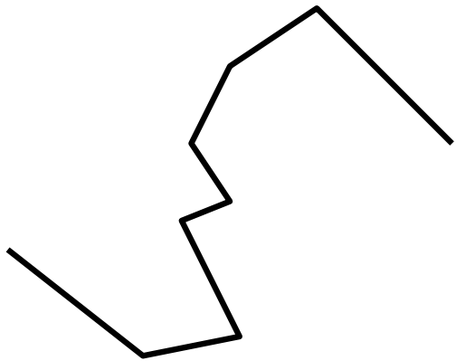
Criteria.
- minimize number of links **(R3)**

Path-Based Schematization

Goal. Solve restricted problem, where G is a path (or polyline)

Constraints.
- $\mathcal{C}$-oriented edges (e.g. octilinear) **(R2)**
- bounded displacement **(R6)**

Criteria.
- minimize number of links **(R3)**

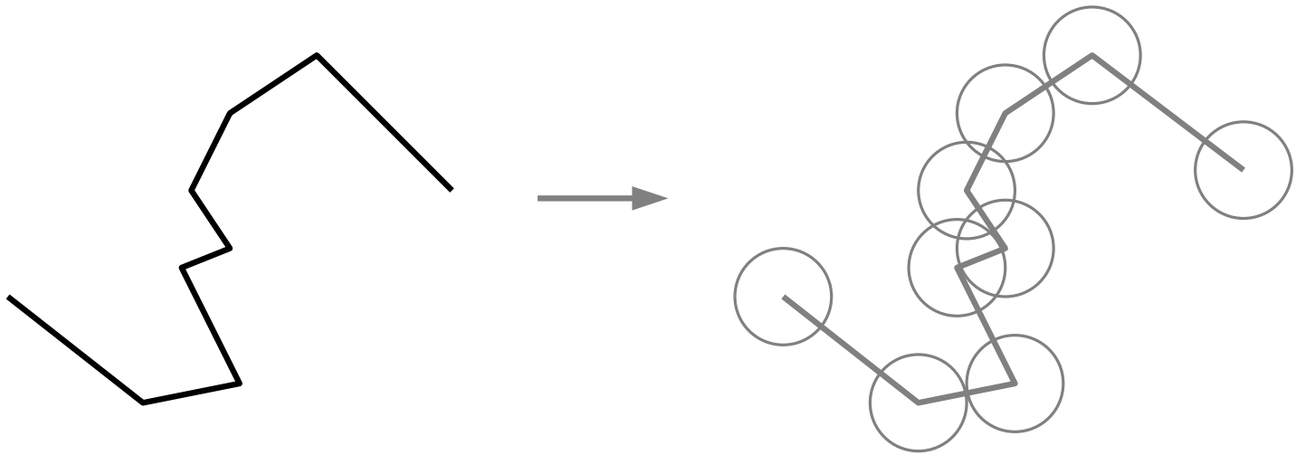


Path-Based Schematization

Goal. Solve restricted problem, where G is a path (or polyline)

Constraints.
- $\mathcal{C}$-oriented edges (e.g. octilinear) **(R2)**
- bounded displacement **(R6)**

Criteria.
- minimize number of links **(R3)**

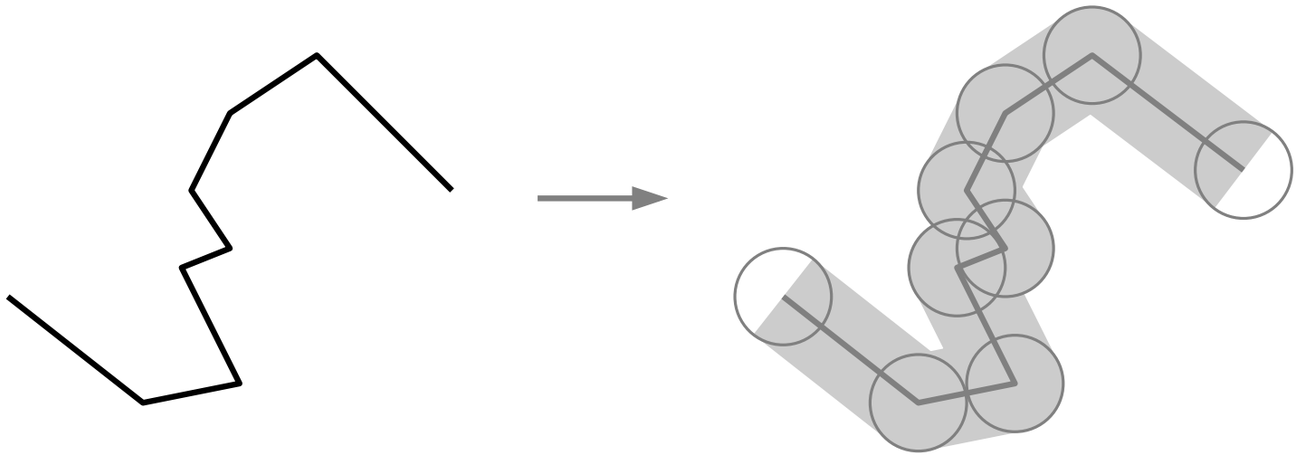


Path-Based Schematization

Goal. Solve restricted problem, where G is a path (or polyline)

Constraints.
- $\mathcal{C}$-oriented edges (e.g. octilinear) **(R2)**
- bounded displacement **(R6)**

Criteria.
- minimize number of links **(R3)**

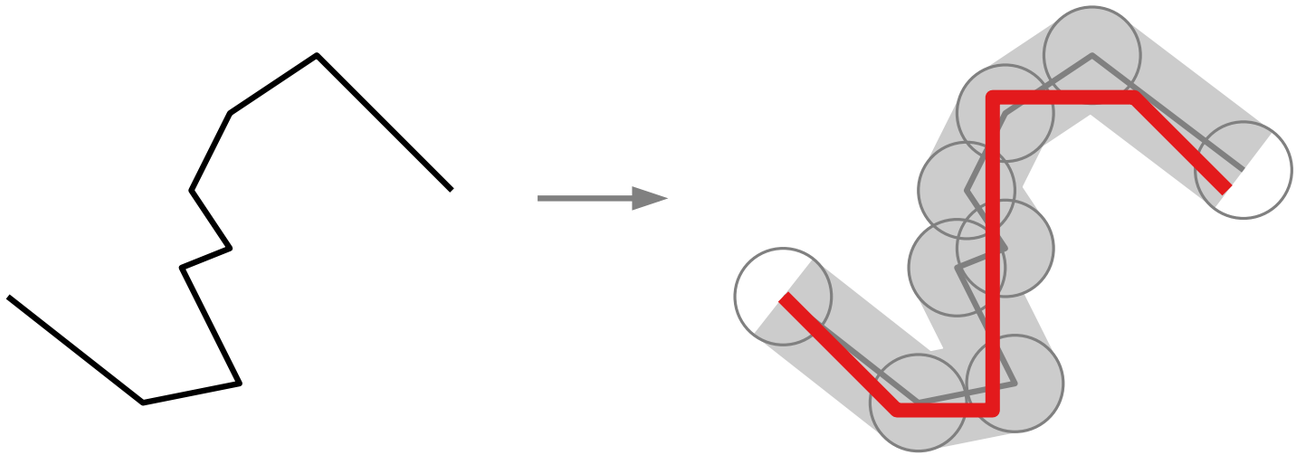


Path-Based Schematization

Goal. Solve restricted problem, where G is a path (or polyline)

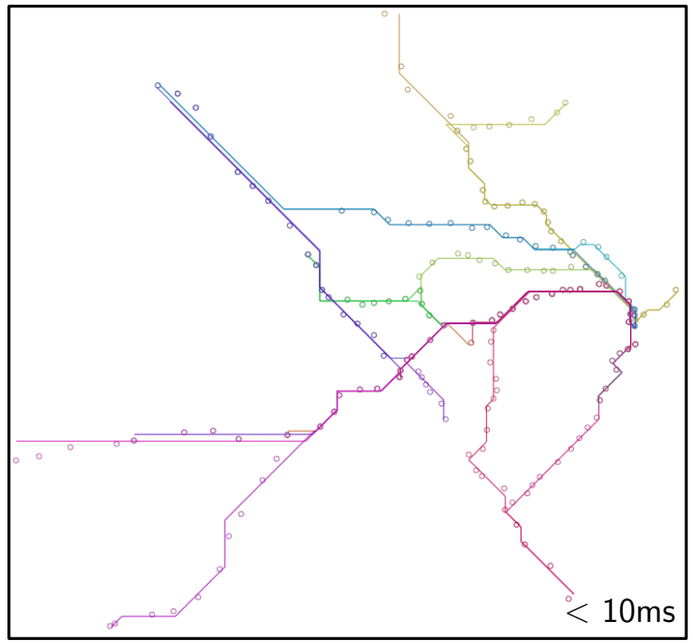
Constraints.
- $\mathcal{C}$-oriented edges (e.g. octilinear) **(R2)**
- bounded displacement **(R6)**

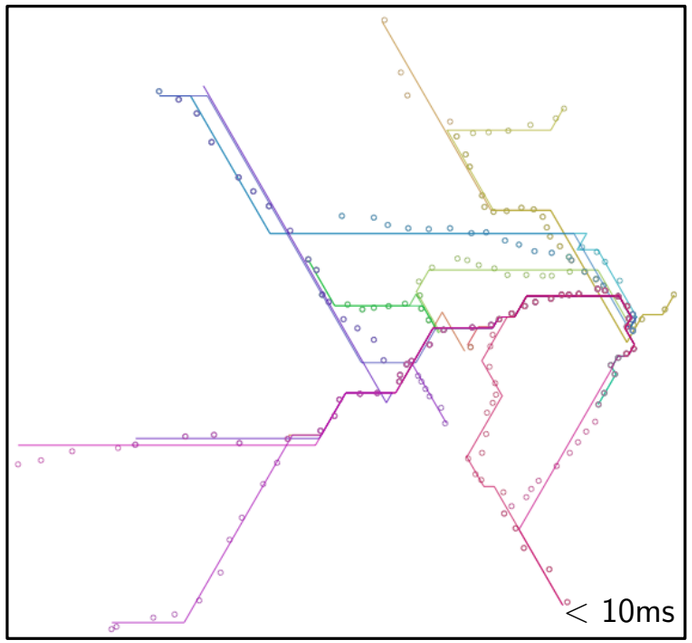
Criteria.
- minimize number of links **(R3)**



Path-based schematization – example

Theorem 2. [Dwyer, Hurst, Merrick '08]
For a path P of length n and orientation set $\mathcal{C}$ a $\mathcal{C}$-oriented schematized path can heuristically be fitted to the vertices in $O(|\mathcal{C}|n)$ time (or $O(|\mathcal{C}|n \log n)$) using least-squares regression.





$\mathcal{C}$-oriented Route Sketches

Theorem 3. [Delling et al. 2010]

Given a monotone path P and a set $\mathcal{C}$ of admissible edge slopes, we can compute, in $O(n^2)$+solve(LP) time, a $\mathcal{C}$-oriented schematization of P, which

- preserves the orthogonal order of P,
- has minimum slope deviation,
- has minimum total length.

$\mathcal{C}$-oriented Route Sketches

Theorem 3. [Delling et al. 2010]

Given a monotone path P and a set $\mathcal{C}$ of admissible edge slopes, we can compute, in $O(n^2)$+solve(LP) time, a $\mathcal{C}$-oriented schematization of P, which

- preserves the orthogonal order of P,
- has minimum slope deviation,
- has minimum total length.

relative north-south-east-west relationship of all vertices

$\mathcal{C}$-oriented Route Sketches

Theorem 3. [Delling et al. 2010]

Given a monotone path P and a set $\mathcal{C}$ of admissible edge slopes, we can compute, in $O(n^2)$+solve(LP) time, a $\mathcal{C}$-oriented schematization of P, which

- preserves the orthogonal order of P,
- has minimum slope deviation,
- has minimum total length.

relative north-south-east-west relationship of all vertices

Proof.

- dynamic programming for slope assignment
- LP for length assignment

$\mathcal{C}$-oriented Route Sketches

Theorem 3. [Delling et al. 2010]

Given a monotone path P and a set $\mathcal{C}$ of admissible edge slopes, we can compute, in $O(n^2)$+solve(LP) time, a $\mathcal{C}$-oriented schematization of P, which

- preserves the orthogonal order of P,
- has minimum slope deviation,
- has minimum total length.

relative north-south-east-west relationship of all vertices

Proof.

- dynamic programming for slope assignment
- LP for length assignment

Theorem 4. [Brandes & Pampel 2009, Gemsa et al. 2011]

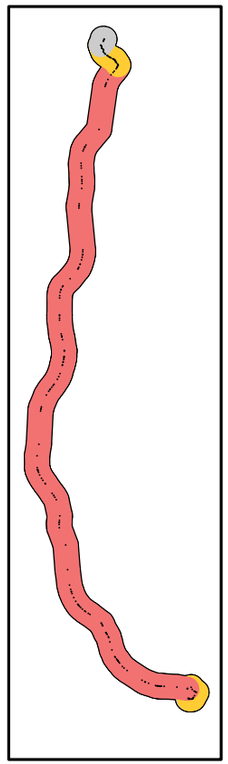
The d-regular (non-monotone) route sketch problem is NP-hard for any $d \geq 1$, where $\mathcal{C} = \{i \cdot 90°/d \mid i \in \mathbb{Z}\}$.

$\mathcal{C}$-oriented Route Sketches

Example. Bremen to Cuxhaven [Gemsa et al. 2011]

$\mathcal{C}$-oriented Route Sketches

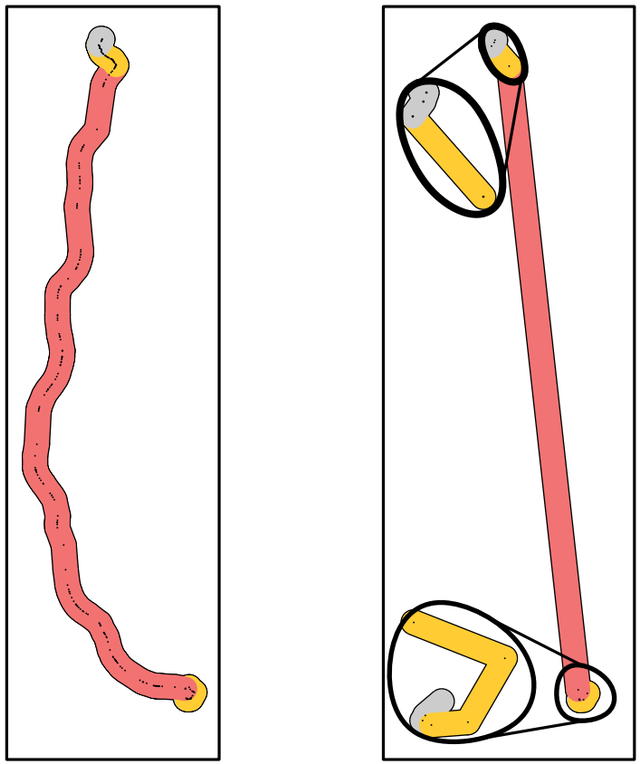
Example. Bremen to Cuxhaven [Gemsa et al. 2011]



input

$\mathcal{C}$-oriented Route Sketches

Example. Bremen to Cuxhaven [Gemsa et al. 2011]

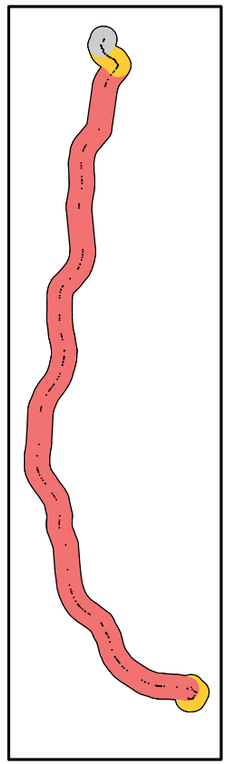


input

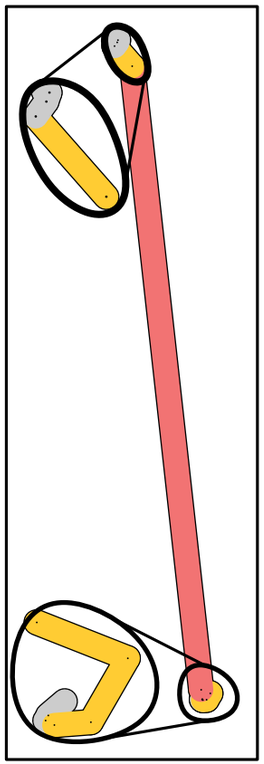
simplified

$\mathcal{C}$-oriented Route Sketches

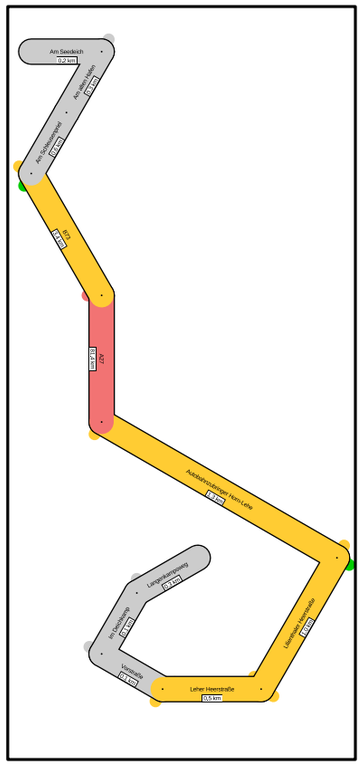
Example. Bremen to Cuxhaven [Gemsa et al. 2011]



input



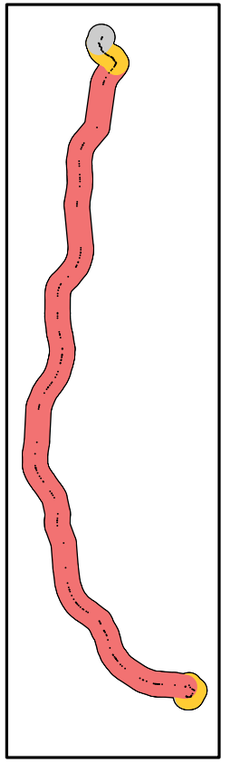
simplified



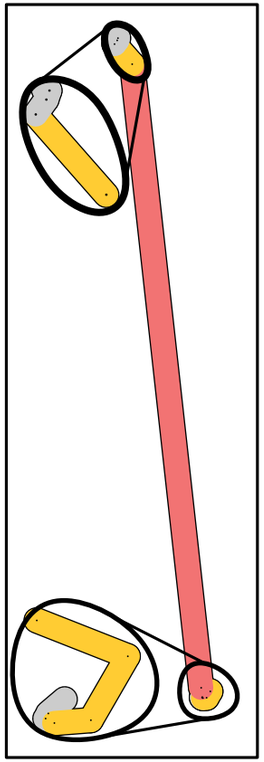
optimized by MIP

$\mathcal{C}$-oriented Route Sketches

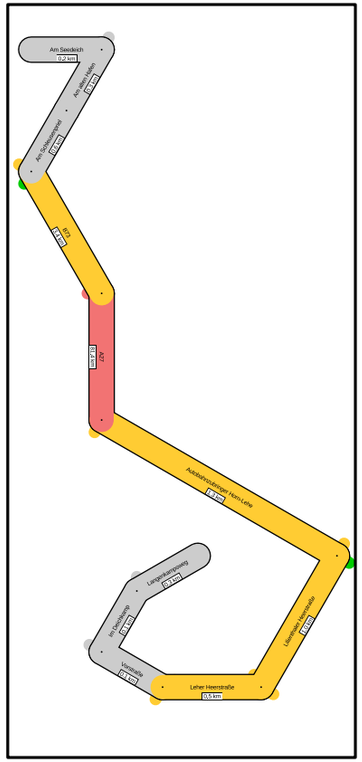
Example. Bremen to Cuxhaven [Gemsa et al. 2011]



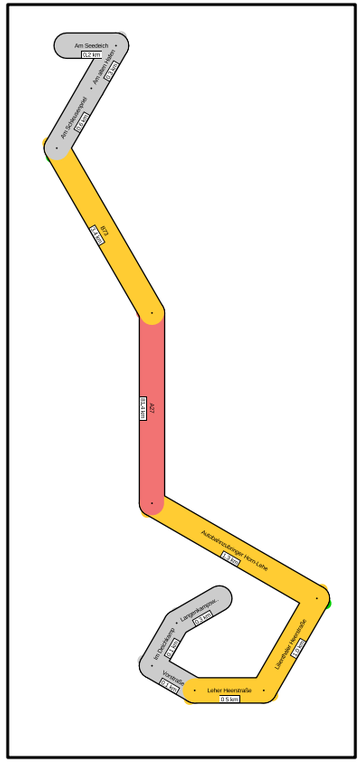
input



simplified



optimized by MIP



including length order constraint

Path-Based Schematization – Discussion

Pros.

Cons.

Path-Based Schematization – Discussion

Pros.

- polynomial running times

Cons.

Path-Based Schematization – Discussion

Pros.

- polynomial running times
- $\mathcal{C}$-orientation and bounded displacement guaranteed

Cons.

Path-Based Schematization – Discussion

Pros.

- polynomial running times
- $\mathcal{C}$-orientation and bounded displacement guaranteed
- bend minimization

Cons.

Path-Based Schematization – Discussion

Pros.

- polynomial running times
- $\mathcal{C}$-orientation and bounded displacement guaranteed
- bend minimization
- extends to metro networks:

Cons.

Path-Based Schematization – Discussion

Pros.

- polynomial running times
- $\mathcal{C}$-orientation and bounded displacement guaranteed
- bend minimization
- extends to metro networks:
  - decompose metro network into paths

Cons.

Path-Based Schematization – Discussion

Pros.

- polynomial running times
- $\mathcal{C}$-orientation and bounded displacement guaranteed
- bend minimization
- extends to metro networks:
  - decompose metro network into paths
  - schematize individual paths

Cons.

Path-Based Schematization – Discussion

Pros.

- polynomial running times
- $\mathcal{C}$-orientation and bounded displacement guaranteed
- bend minimization
- extends to metro networks:
  - decompose metro network into paths
  - schematize individual paths
  - glue schematized paths together at interchanges

Cons.

Path-Based Schematization – Discussion

Pros.

- polynomial running times
- $\mathcal{C}$-orientation and bounded displacement guaranteed
- bend minimization
- extends to metro networks:
  - decompose metro network into paths
  - schematize individual paths
  - glue schematized paths together at interchanges

Cons.

- no guarantee on network topology **(R1)**

Path-Based Schematization – Discussion

Pros.

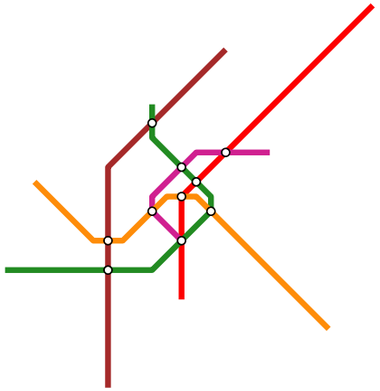
- polynomial running times
- $\mathcal{C}$-orientation and bounded displacement guaranteed
- bend minimization
- extends to metro networks:
  - decompose metro network into paths
  - schematize individual paths
  - glue schematized paths together at interchanges

Cons.

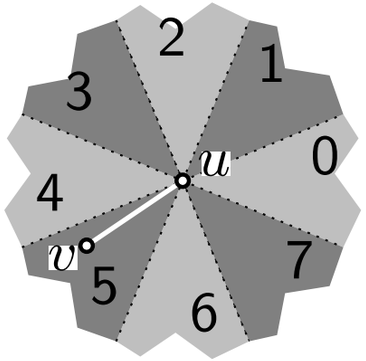
- no guarantee on network topology **(R1)**
- distortion/displacement too limited for metro maps

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

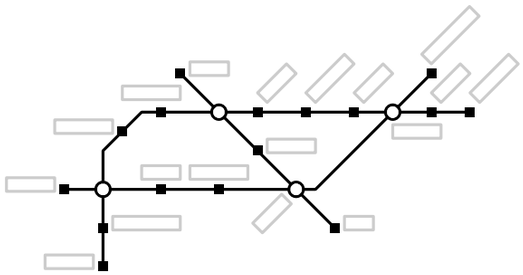
Visualization of Graphs



Lecture 12:
Octilinear Graph Drawing
Metro Map Layout



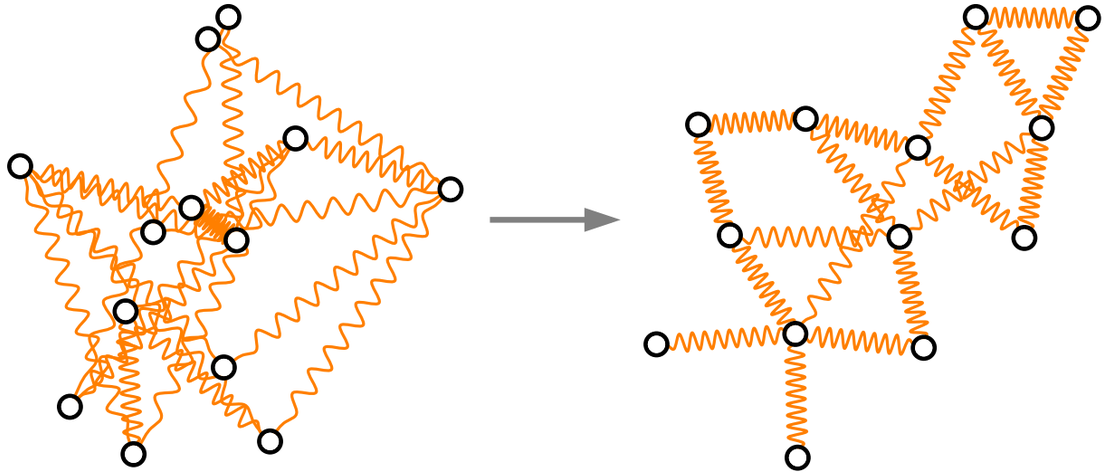
Part III:
Force-Based Schematization



Alexander Wolff

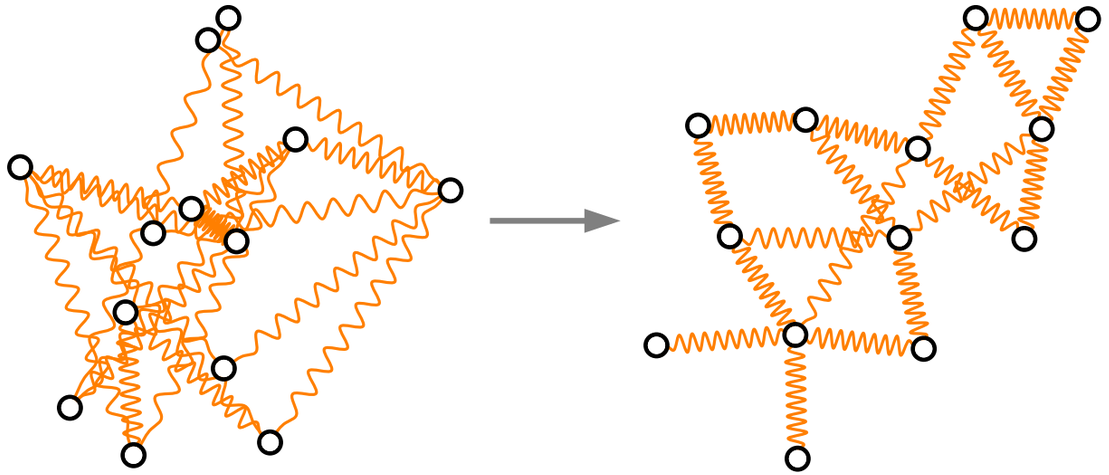
Force-Based Schematization

Idea. Apply well known force-based graph drawing approach



Force-Based Schematization

Idea. Apply well known force-based graph drawing approach

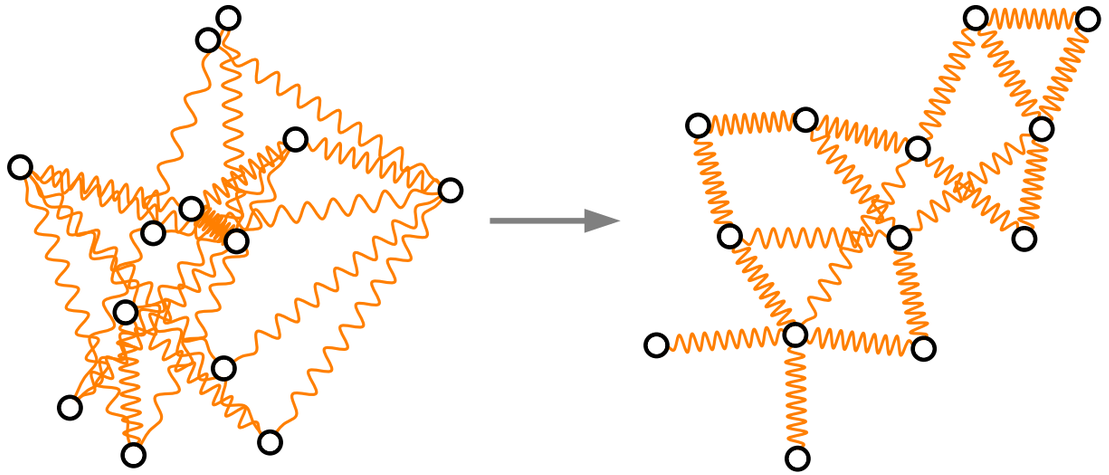


Recall.

- vertices are charged particles repelling each other

Force-Based Schematization

Idea. Apply well known force-based graph drawing approach

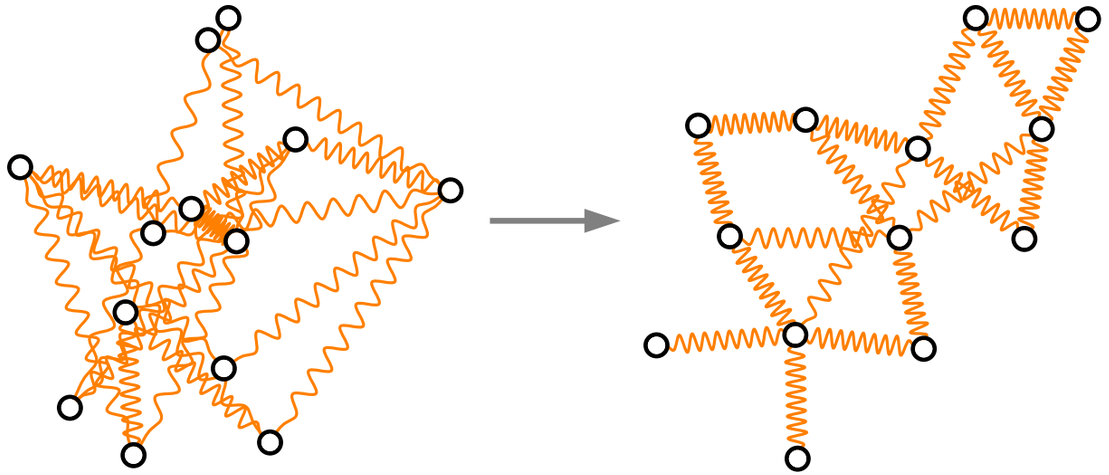


Recall.

- vertices are charged particles repelling each other
- edges are springs pulling edges into target length

Force-Based Schematization

Idea. Apply well known force-based graph drawing approach

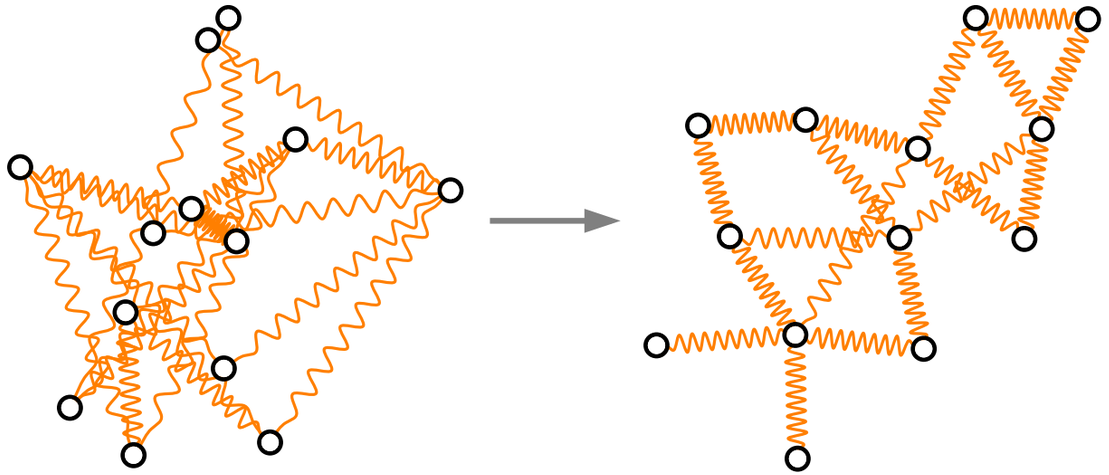


Recall.

- vertices are charged particles repelling each other
- edges are springs pulling edges into target length
- iteratively calculate and apply forces until system stabilizes

Force-Based Schematization

Idea. Apply well known force-based graph drawing approach



Recall.

- vertices are charged particles repelling each other
- edges are springs pulling edges into target length
- iteratively calculate and apply forces until system stabilizes

$\rightarrow$ define additional forces to model subset of metro map design rules

Force-Based Schematization – Octilinear

[Hong et al. 2006]

- contract degree-2 vertices into weighted edges **(R3)**

Force-Based Schematization – Octilinear

[Hong et al. 2006]

- contract degree-2 vertices into weighted edges **(R3)**
- define octilinear magnetic field attracting edges **(R2)**

Force-Based Schematization – Octilinear

[Hong et al. 2006]

- contract degree-2 vertices into weighted edges **(R3)**
- define octilinear magnetic field attracting edges **(R2)**
- only apply topology preserving vertex moves **(R1)**

Force-Based Schematization – Octilinear

[Hong et al. 2006]

- contract degree-2 vertices into weighted edges **(R3)**
- define octilinear magnetic field attracting edges **(R2)**
- only apply topology preserving vertex moves **(R1)**
- spring lengths model uniform edge lengths **(R7)**

Force-Based Schematization – Octilinear

[Hong et al. 2006]

- contract degree-2 vertices into weighted edges **(R3)**
- define octilinear magnetic field attracting edges **(R2)**
- only apply topology preserving vertex moves **(R1)**
- spring lengths model uniform edge lengths **(R7)**
- vertex repulsion models feature separation **(R8)**

Force-Based Schematization – Octilinear

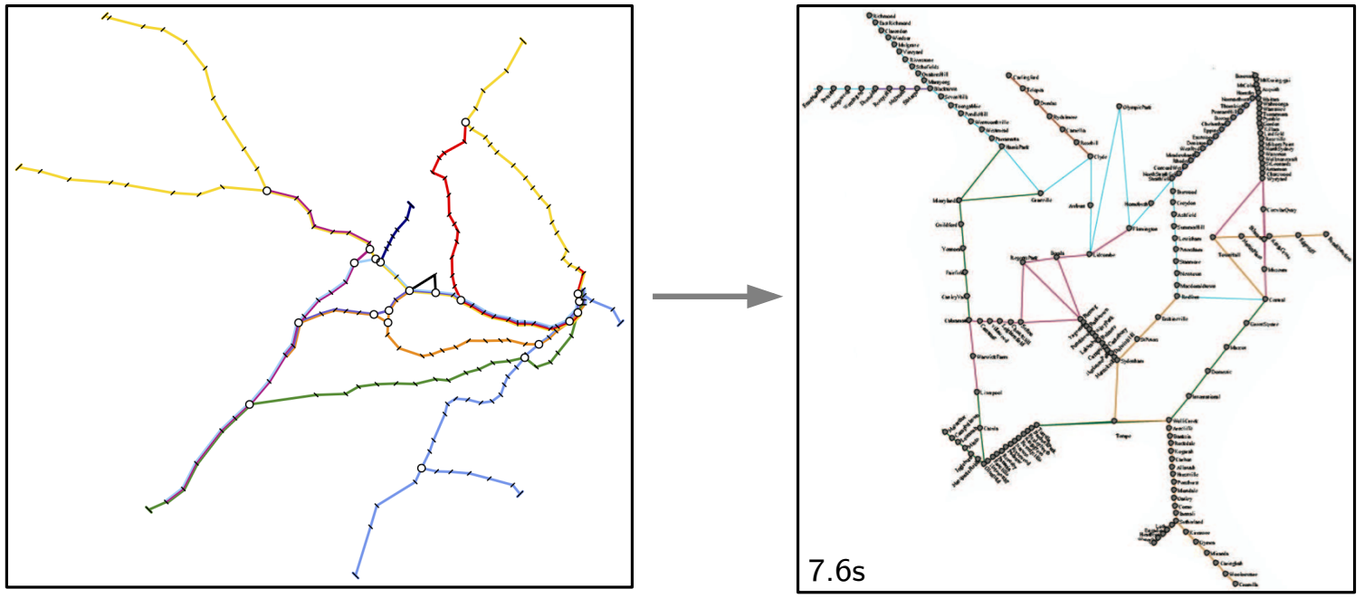
[Hong et al. 2006]

- contract degree-2 vertices into weighted edges **(R3)**
- define octilinear magnetic field attracting edges **(R2)**
- only apply topology preserving vertex moves **(R1)**
- spring lengths model uniform edge lengths **(R7)**
- vertex repulsion models feature separation **(R8)**
- station labels placed in independent 2nd step

Force-Based Schematization – Octilinear

[Hong et al. 2006]

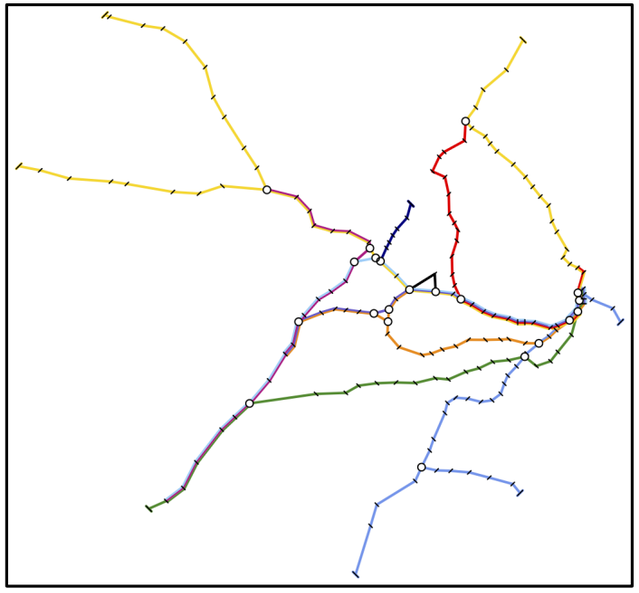
- contract degree-2 vertices into weighted edges **(R3)**
- define octilinear magnetic field attracting edges **(R2)**
- only apply topology preserving vertex moves **(R1)**
- spring lengths model uniform edge lengths **(R7)**
- vertex repulsion models feature separation **(R8)**
- station labels placed in independent 2nd step

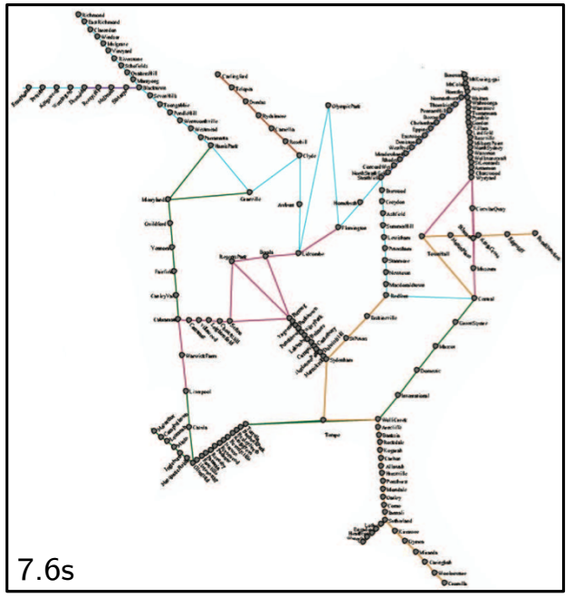


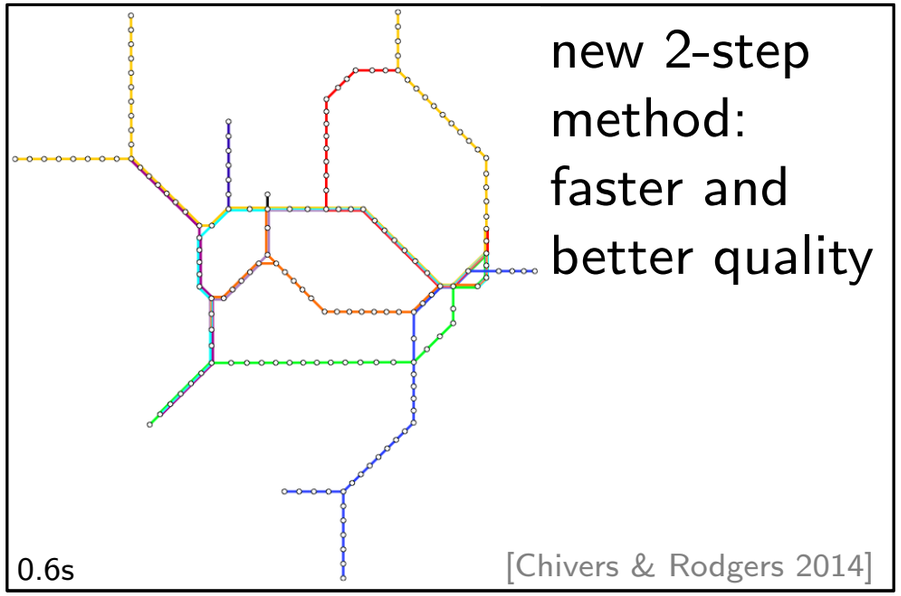
Force-Based Schematization – Octilinear

[Hong et al. 2006]

- contract degree-2 vertices into weighted edges **(R3)**
- define octilinear magnetic field attracting edges **(R2)**
- only apply topology preserving vertex moves **(R1)**
- spring lengths model uniform edge lengths **(R7)**
- vertex repulsion models feature separation **(R8)**
- station labels placed in independent 2nd step







Force-Based Schematization – Bézier

[Fink et al. 2013]

- convert (octilinear) input layout into Bézier curves
  $\rightarrow$ vertices and control points, both subject to forces

Force-Based Schematization – Bézier

[Fink et al. 2013]

- convert (octilinear) input layout into Bézier curves
  $\rightarrow$ vertices and control points, both subject to forces
- metro line subcurves share tangents in common vertex **(R4)**

Force-Based Schematization – Bézier

[Fink et al. 2013]

- convert (octilinear) input layout into Bézier curves
  $\rightarrow$ vertices and control points, both subject to forces
- metro line subcurves share tangents in common vertex **(R4)**
- (standard) attractive and repulsive forces **(R7)+(R8)**

Force-Based Schematization – Bézier

[Fink et al. 2013]

- convert (octilinear) input layout into Bézier curves
  $\rightarrow$ vertices and control points, both subject to forces
- metro line subcurves share tangents in common vertex **(R4)**
- (standard) attractive and repulsive forces **(R7)+(R8)**
- (weak) force towards initial position **(R6)**

Force-Based Schematization – Bézier

[Fink et al. 2013]

- convert (octilinear) input layout into Bézier curves
  $\rightarrow$ vertices and control points, both subject to forces
- metro line subcurves share tangents in common vertex **(R4)**
- (standard) attractive and repulsive forces **(R7)+(R8)**
- (weak) force towards initial position **(R6)**
- forces improving curve shape (low curvature) **(R3)**

Force-Based Schematization – Bézier

[Fink et al. 2013]

- convert (octilinear) input layout into Bézier curves
  $\rightarrow$ vertices and control points, both subject to forces
- metro line subcurves share tangents in common vertex **(R4)**
- (standard) attractive and repulsive forces **(R7)+(R8)**
- (weak) force towards initial position **(R6)**
- forces improving curve shape (low curvature) **(R3)**
- merge curves whenever possible **(R3)**

Force-Based Schematization – Bézier

[Fink et al. 2013]

- convert (octilinear) input layout into Bézier curves
  $\rightarrow$ vertices and control points, both subject to forces
- metro line subcurves share tangents in common vertex **(R4)**
- (standard) attractive and repulsive forces **(R7)+(R8)**
- (weak) force towards initial position **(R6)**
- forces improving curve shape (low curvature) **(R3)**
- merge curves whenever possible **(R3)**
- forces improving angular resolution **(R5)**

Force-Based Schematization – Bézier

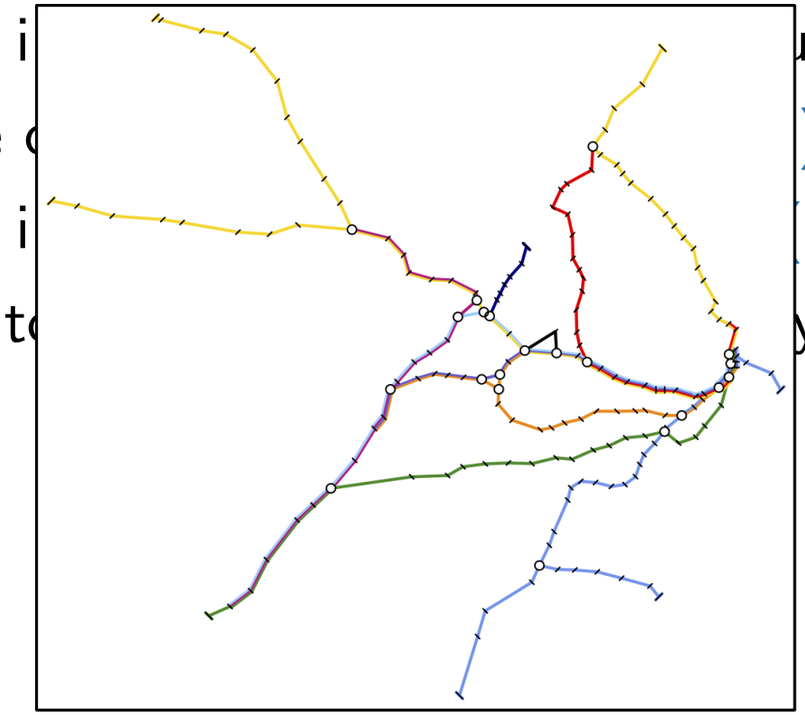
[Fink et al. 2013]

- convert (octilinear) input layout into Bézier curves
  $\rightarrow$ vertices and control points, both subject to forces
- metro line subcurves share tangents in common vertex **(R4)**
- (standard) attractive and repulsive forces **(R7)+(R8)**
- (weak) force towards initial position **(R6)**
- forces improving curve shape (low curvature) **(R3)**
- merge curves whenever possible **(R3)**
- forces improving angular resolution **(R5)**
- apply topology-preserving moves only **(R1)**

Force-Based Schematization – Bézier

[Fink et al. 2013]

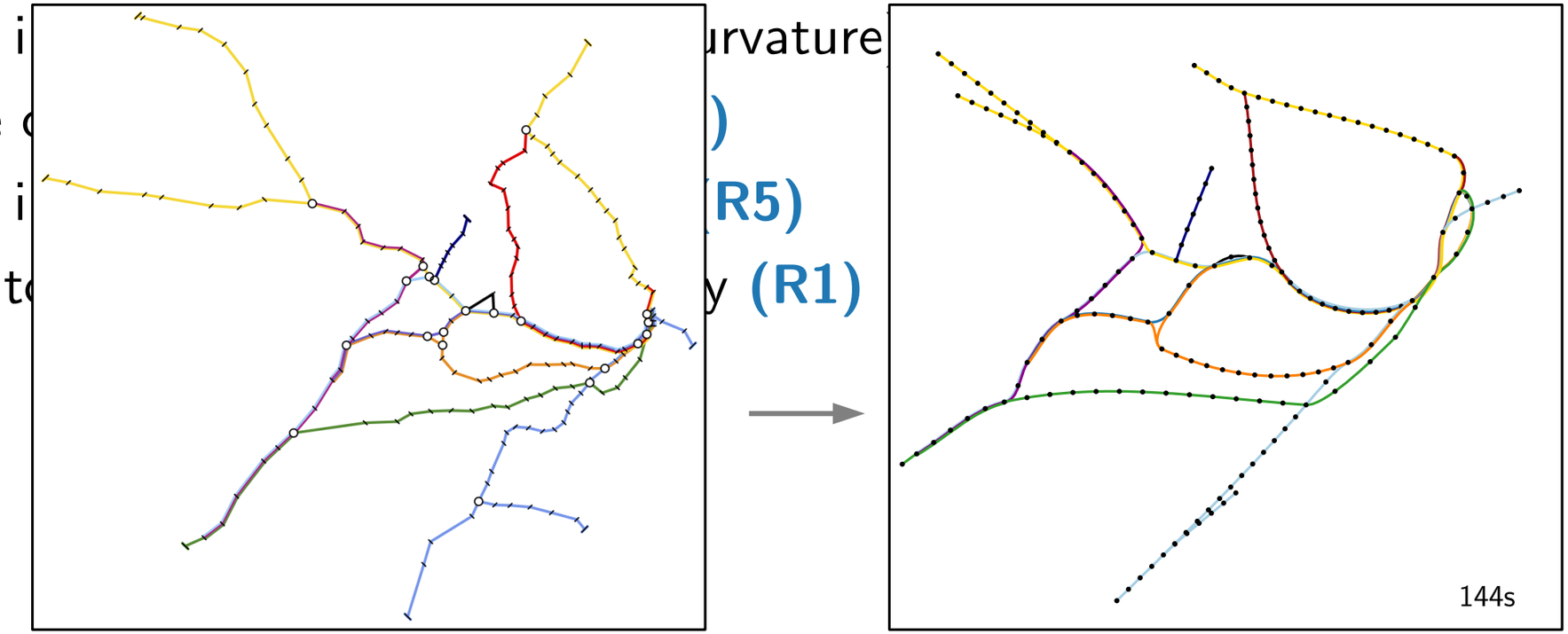
- convert (octilinear) input layout into Bézier curves
  $\rightarrow$ vertices and control points, both subject to forces
- metro line subcurves share tangents in common vertex **(R4)**
- (standard) attractive and repulsive forces **(R7)+(R8)**
- (weak) force towards initial position **(R6)**
- forces i urvature) **(R3)**
- merge)
- forces i (R5)
- apply t y **(R1)**



Force-Based Schematization – Bézier

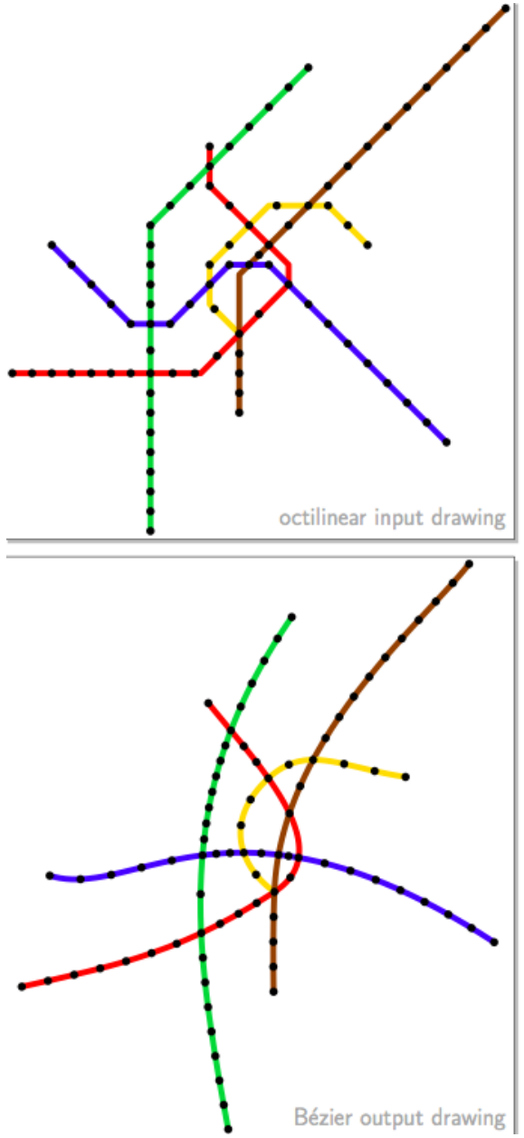
[Fink et al. 2013]

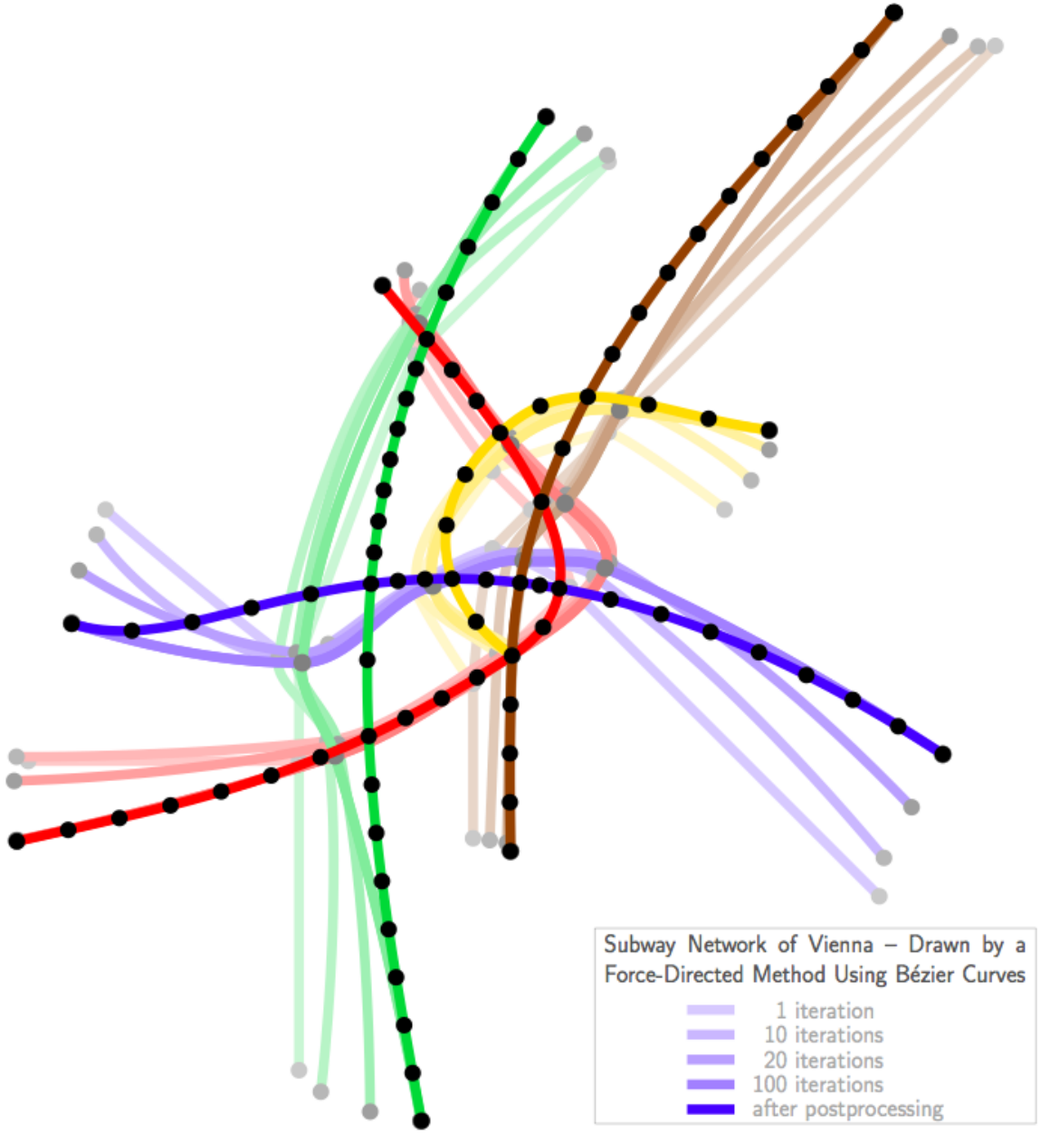
- convert (octilinear) input layout into Bézier curves
  $\rightarrow$ vertices and control points, both subject to forces
- metro line subcurves share tangents in common vertex **(R4)**
- (standard) attractive and repulsive forces **(R7)+(R8)**
- (weak) force towards initial position **(R6)**
- forces i… …urvature…
- merge … **)**
- forces i… **(R5)**
- apply t… …y **(R1)**



Force-Based Schematization – Bézier

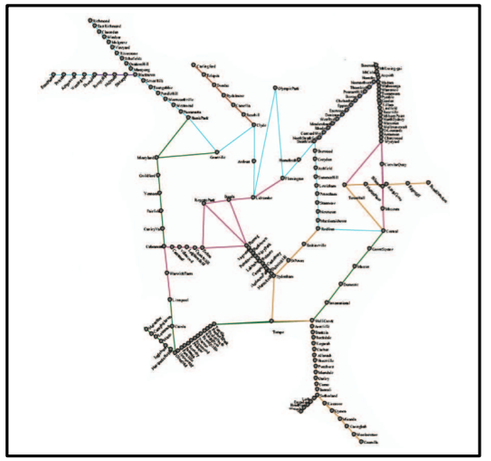
[Fink et al. 2013]



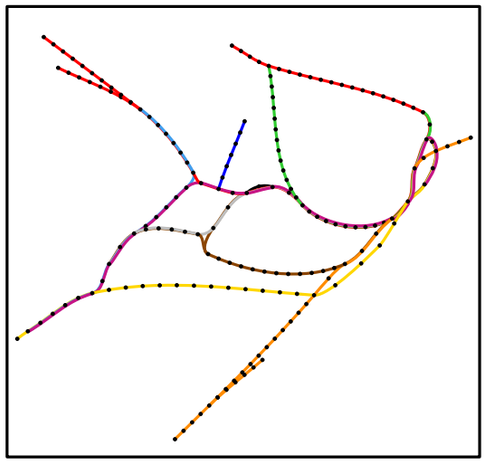


Force-Based Schematization – Discussion

Octilinear.



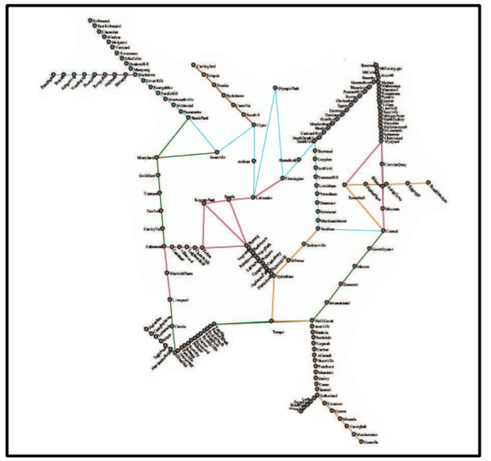
Bézier.

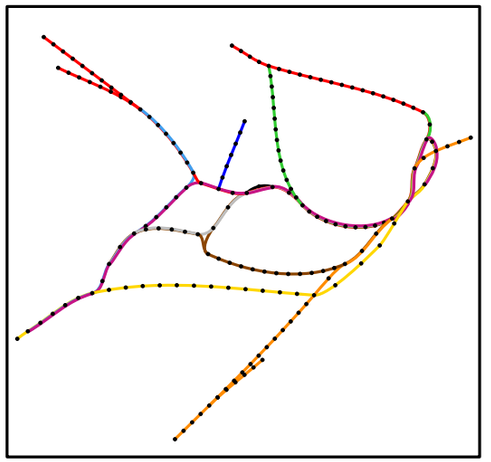


Force-Based Schematization – Discussion

Octilinear.

- guarantees topology **(R1)**
- slower than path-based algorithms, but still fast enough
- no strict enforcement of octilinearity
- quite unbalanced edge lengths
- bends in interchanges
- no distortion restriction



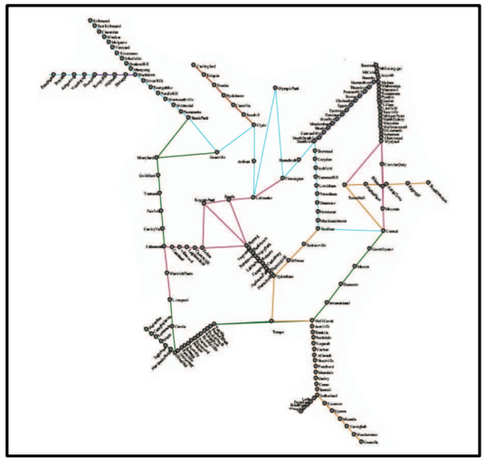


Bézier.

Force-Based Schematization – Discussion

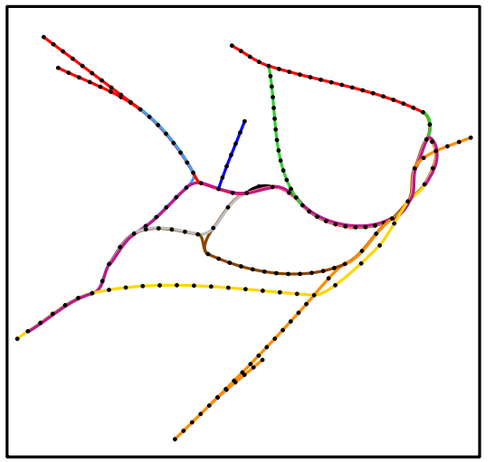
Octilinear.

- guarantees topology **(R1)**
- slower than path-based algorithms, but still fast enough
- no strict enforcement of octilinearity
- quite unbalanced edge lengths
- bends in interchanges
- no distortion restriction



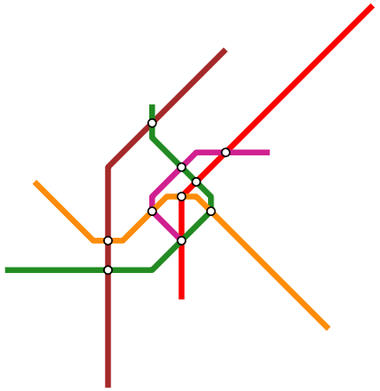
Bézier.

- guarantees topology **(R1)**
- takes almost all design rules into account
- first curvilinear metro map algorithm
- works well on small and medium instances
- difficulties with more complex networks

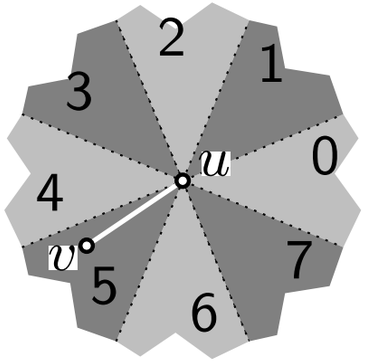


Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

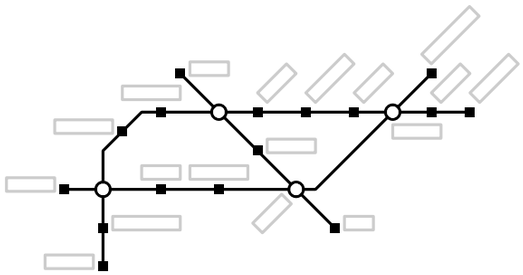
Visualization of Graphs



Lecture 12:
Octilinear Graph Drawing
Metro Map Layout



Part IV:
Local Schematization



Alexander Wolff

Local Schematization

Idea.

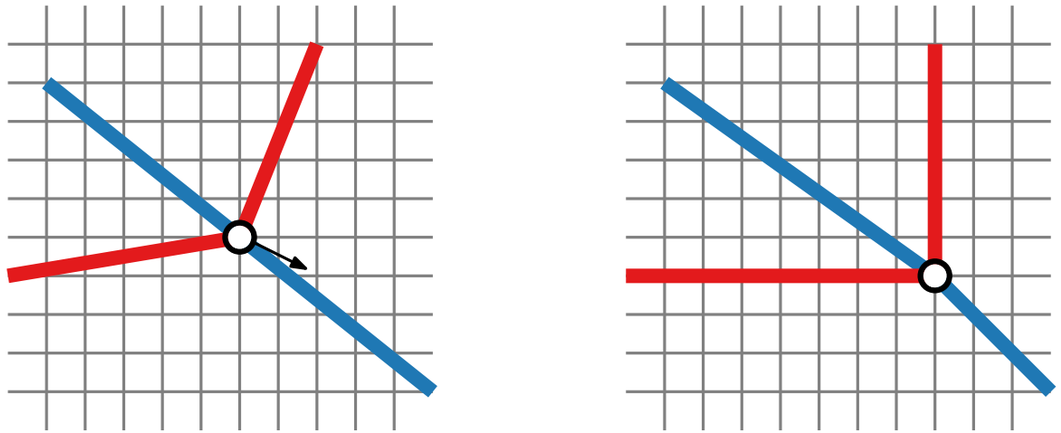
- define a (multi-criteria) layout quality function

es

Local Schematization

Idea.

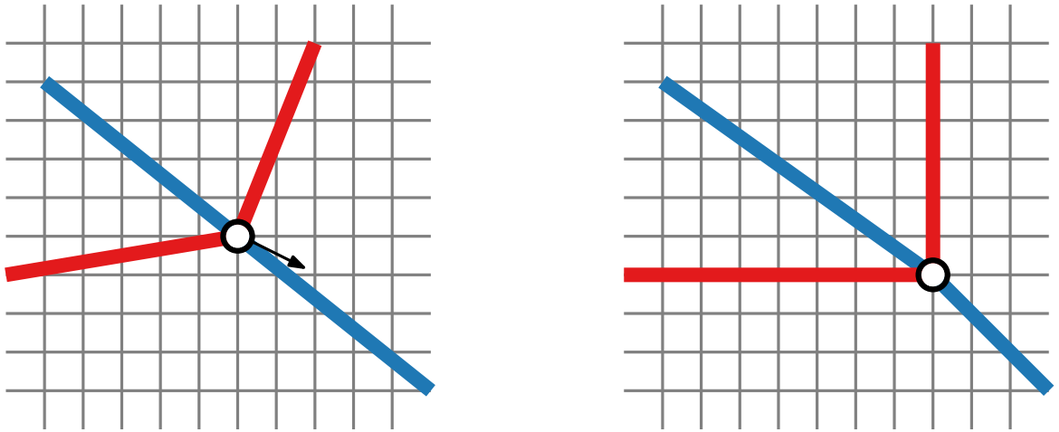
- define a (multi-criteria) layout quality function
- improve layout quality step-by-step by locally moving vertices to better nearby (grid) positions



Local Schematization

Idea.

- define a (multi-criteria) layout quality function
- improve layout quality step-by-step by locally moving vertices to better nearby (grid) positions
- applicable optimization techniques: hill climbing, simulated annealing, ant colony optimization, ...



Local Schematization

Idea.

- define a (multi-criteria) layout quality function
- improve layout quality step-by-step by locally moving vertices to better nearby (grid) positions
- applicable optimization techniques: hill climbing, simulated annealing, ant colony optimization, ...

[Avelar & Müller 2000]

- calculate best vertex position in each criterion (octilinearity **(R2)**, min. separation **(R8)**)

Local Schematization

Idea.

- define a (multi-criteria) layout quality function
- improve layout quality step-by-step by locally moving vertices to better nearby (grid) positions
- applicable optimization techniques: hill climbing, simulated annealing, ant colony optimization, ...

[Avelar & Müller 2000]

- calculate best vertex position in each criterion (octilinearity **(R2)**, min. separation **(R8)**)
- move vertex to average of positions without violating topology **(R1)**

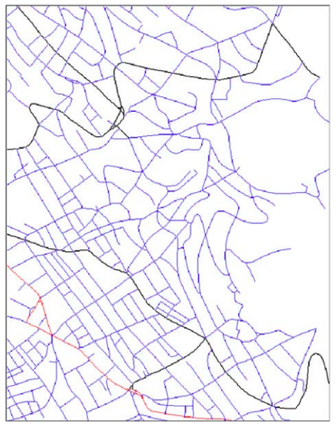
Local Schematization

Idea.

- define a (multi-criteria) layout quality function
- improve layout quality step-by-step by locally moving vertices to better nearby (grid) positions
- applicable optimization techniques: hill climbing, simulated annealing, ant colony optimization, ...

[Avelar & Müller 2000]

- calculate best vertex position in each criterion (octilinearity **(R2)**, min. separation **(R8)**)
- move vertex to average of positions without violating topology **(R1)**



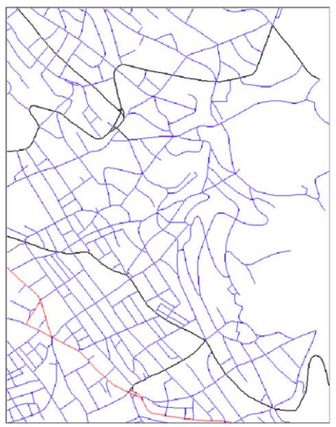
Local Schematization

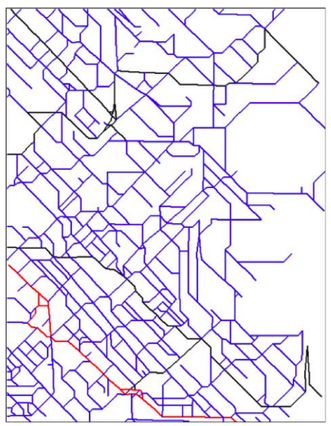
Idea.

- define a (multi-criteria) layout quality function
- improve layout quality step-by-step by locally moving vertices to better nearby (grid) positions
- applicable optimization techniques: hill climbing, simulated annealing, ant colony optimization, ...

[Avelar & Müller 2000]

- calculate best vertex position in each criterion (octilinearity **(R2)**, min. separation **(R8)**)
- move vertex to average of positions without violating topology **(R1)**





100

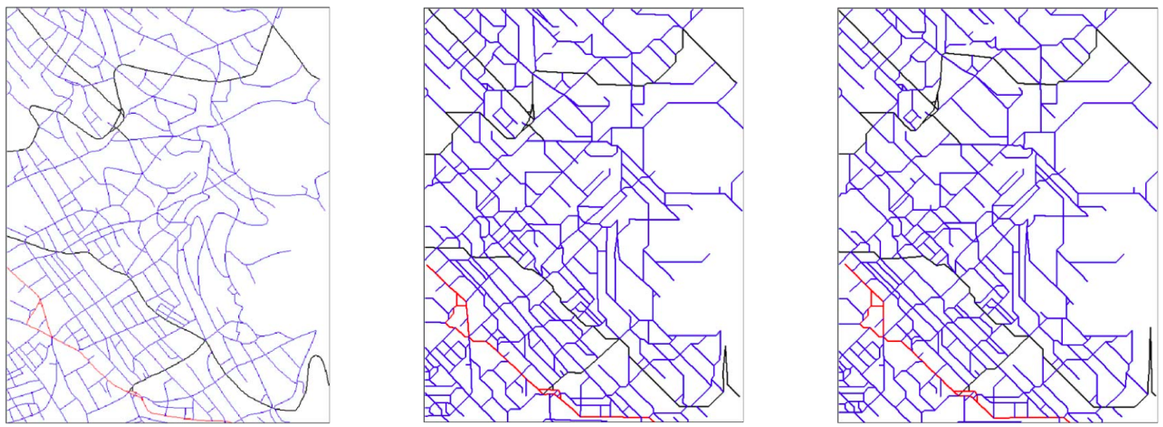
Local Schematization

Idea.

- define a (multi-criteria) layout quality function
- improve layout quality step-by-step by locally moving vertices to better nearby (grid) positions
- applicable optimization techniques: hill climbing, simulated annealing, ant colony optimization, ...

[Avelar & Müller 2000]

- calculate best vertex position in each criterion (octilinearity **(R2)**, min. separation **(R8)**)
- move vertex to average of positions without violating topology **(R1)**



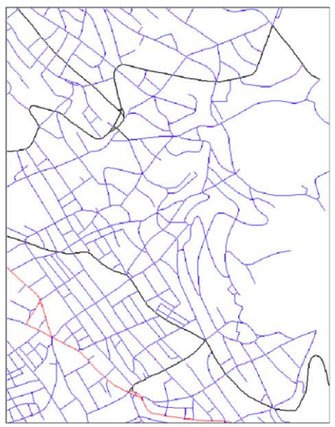
Local Schematization

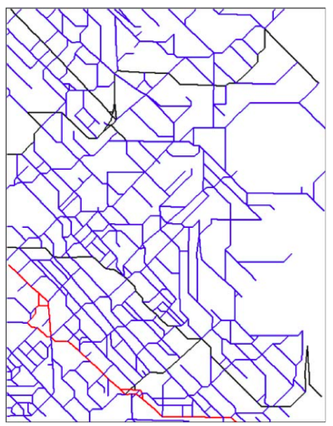
Idea.

- define a (multi-criteria) layout quality function
- improve layout quality step-by-step by locally moving vertices to better nearby (grid) positions
- applicable optimization techniques: hill climbing, simulated annealing, ant colony optimization, . . .

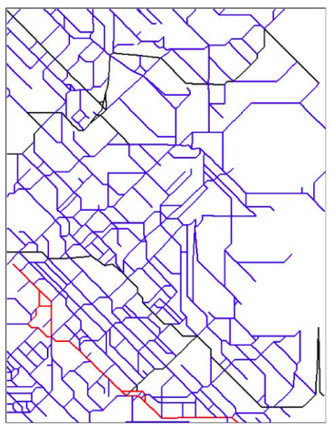
[Avelar & Müller 2000]

- calculate best vertex position in each criterion (octilinearity **(R2)**, min. separation **(R8)**)
- move vertex to average of positions without violating topology **(R1)**





100



1,000



10,000

Local Schematization

Idea.

- define a (multi-criteria) layout quality function
- improve layout quality step-by-step by locally moving vertices to better nearby (grid) positions
- applicable optimization techniques: hill climbing, simulated annealing, ant colony optimization, ...

[Ware et al. 2006, Ware & Richards 2013]

- weighted multicriteria function

Local Schematization

Idea.

- define a (multi-criteria) layout quality function
- improve layout quality step-by-step by locally moving vertices to better nearby (grid) positions
- applicable optimization techniques: hill climbing, simulated annealing, ant colony optimization, ...

[Ware et al. 2006, Ware & Richards 2013]

- weighted multicriteria function
- contract degree-2 vertices prior to optimization

Local Schematization

Idea.

- define a (multi-criteria) layout quality function
- improve layout quality step-by-step by locally moving vertices to better nearby (grid) positions
- applicable optimization techniques: hill climbing, simulated annealing, ant colony optimization, ...

[Ware et al. 2006, Ware & Richards 2013]

- weighted multicriteria function
- contract degree-2 vertices prior to optimization
- implemented more design rules (topology **(R1)**, octilinearity **(R2)**, displacement **(R6)**, edge lengths **(R7)**, separation **(R8)**)

Local Schematization

Idea.

- define a (multi-criteria) layout quality function
- improve layout quality step-by-step by locally moving vertices to better nearby (grid) positions
- applicable optimization techniques: hill climbing, simulated annealing, ant colony optimization, ...

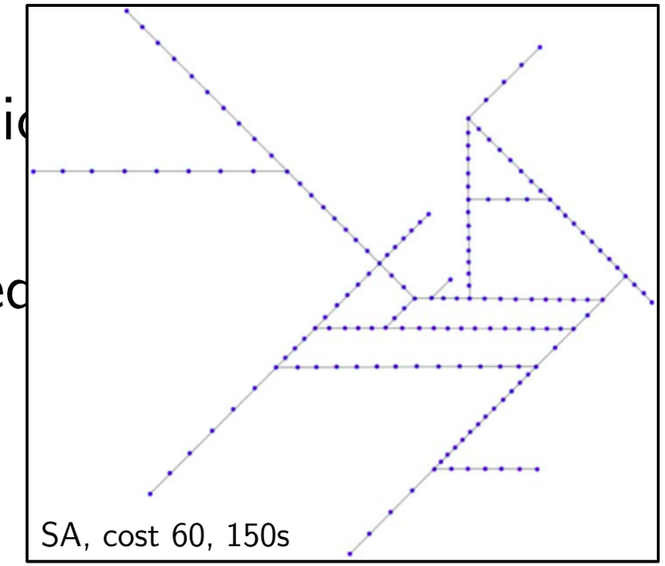
[Ware et al. 2006, Ware & Richards 2013]

- weighted multicriteria function
- contract degree-2 vertices prior to optimization
- implemented more design rules (topology **(R1)**, octilinearity **(R2)**, displacement **(R6)**, edge lengths **(R7)**, separation **(R8)**)
- simulated annealing (2006) and ant colony optimization (2013)

Local Schematization

Idea.

- define a (multi-criteria) layout quality function
- improve layout quality step-by-step by locally moving verti to better nearby (grid) positions
- applicable optimization techniques: hill climbing, simulated annealing, ant colony optimization, ...



[Ware et al. 2006, Ware & Richards 2013]

- weighted multicriteria function
- contract degree-2 vertices prior to optimization
- implemented more design rules (topology **(R1)**, octilinearity **(R2)**, displacement **(R6)**, edge lengths **(R7)**, separation **(R8)**)
- simulated annealing (2006) and ant colony optimization (2013)

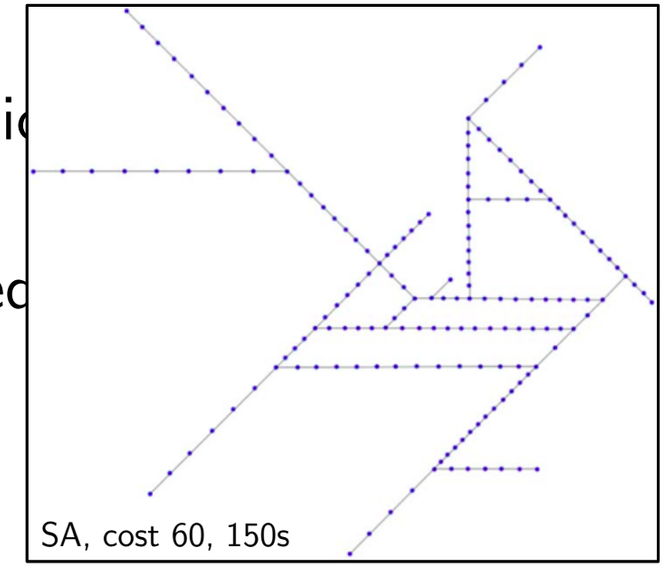
Local Schematization

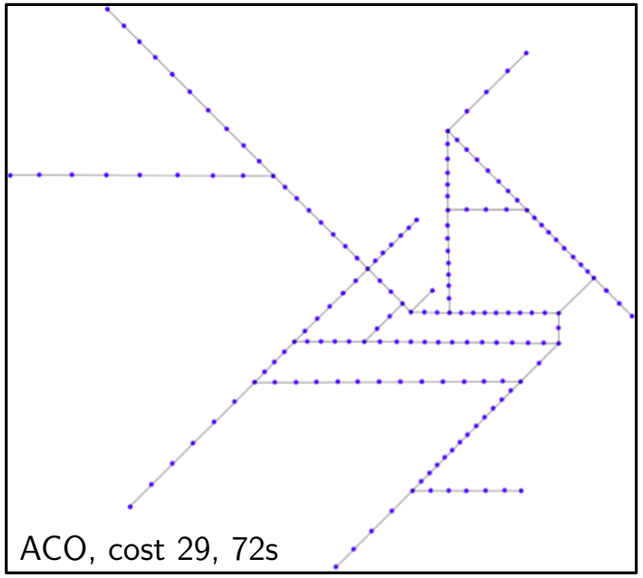
Idea.

- define a (multi-criteria) layout quality function
- improve layout quality step-by-step by locally moving verti to better nearby (grid) positions
- applicable optimization techniques: hill climbing, simulated annealing, ant colony optimization, ...

[Ware et al. 2006, Ware & Richards 2013]

- weighted multicriteria function
- contract degree-2 vertices prior to optimization
- implemented more design rules (topology **(R1)**, octilinearity **(R2)**, displacement **(R6)**, edge lengths **(R7)**, separation **(R8)**)
- simulated annealing (2006) and ant colony optimization (2013)





Local Schematization

[Stott et al. 2011]

Idea.

- design rules as before

Local Schematization

[Stott et al. 2011]

Idea.

- design rules as before
- additionally include metro map specific criteria (bend minimization **(R3)**, interchange straightness **(R4)**, angular resolution **(R5)**, relative positions **(R6)**)

Local Schematization

[Stott et al. 2011]

Idea.

- design rules as before
- additionally include metro map specific criteria (bend minimization **(R3)**, interchange straightness **(R4)**, angular resolution **(R5)**, relative positions **(R6)**)
- integrate alternating label placement rounds

Local Schematization

[Stott et al. 2011]

Idea.

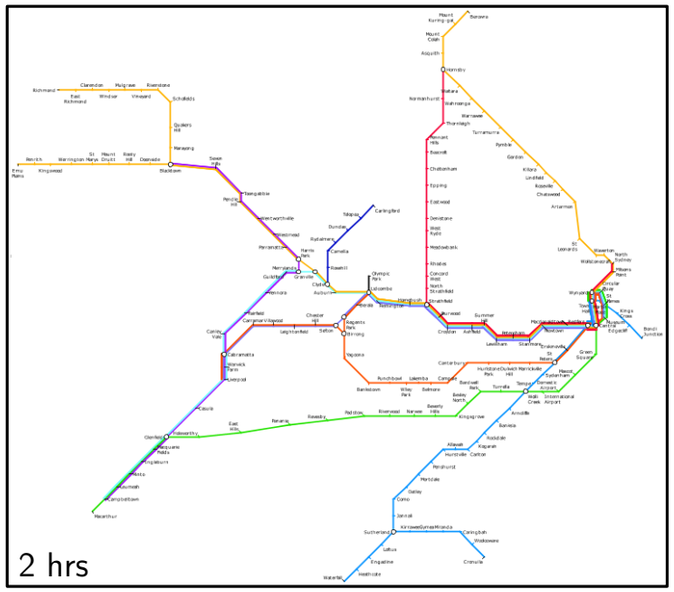
- design rules as before
- additionally include metro map specific criteria (bend minimization **(R3)**, interchange straightness **(R4)**, angular resolution **(R5)**, relative positions **(R6)**)
- integrate alternating label placement rounds
- some ad-hoc fixes for local minima situations

Local Schematization

[Stott et al. 2011]

Idea.

- design rules as before
- additionally include metro map specific criteria (bend minimization **(R3)**, interchange straightness **(R4)**, angular resolution **(R5)**, relative positions **(R6)**)
- integrate alternating label placement rounds
- some ad-hoc fixes for local minima situations

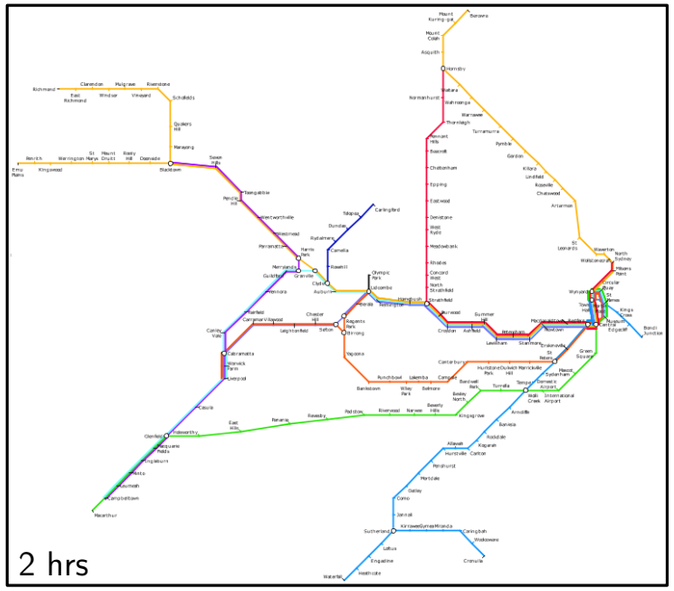


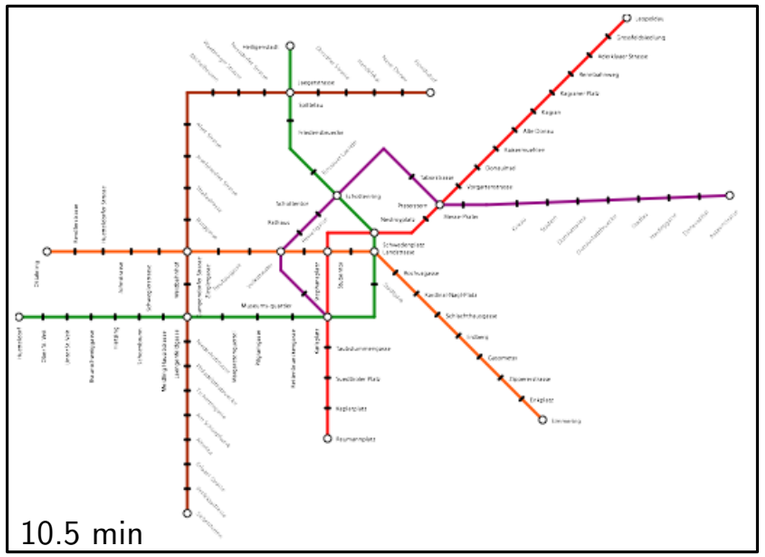
Local Schematization

[Stott et al. 2011]

Idea.

- design rules as before
- additionally include metro map specific criteria (bend minimization **(R3)**, interchange straightness **(R4)**, angular resolution **(R5)**, relative positions **(R6)**)
- integrate alternating label placement rounds
- some ad-hoc fixes for local minima situations





Local Schematization – Discussion

Pros.

Cons.

Local Schematization – Discussion

Pros.

- flexible framework, easy to integrate new criteria
- recent methods improved visual layout quality
- integration of layout and labeling

Cons.

Local Schematization – Discussion

Pros.

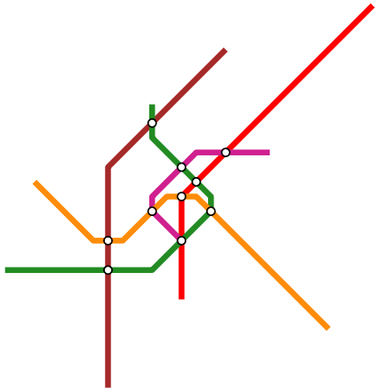
- flexible framework, easy to integrate new criteria
- recent methods improved visual layout quality
- integration of layout and labeling

Cons.

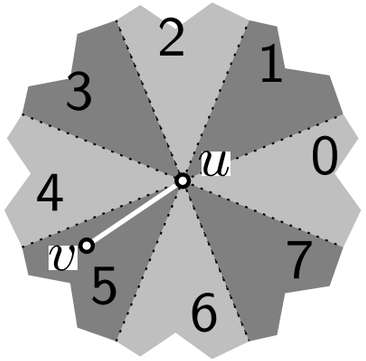
- optimization of criteria, but no guarantees (except topology)
- susceptible to local minima
- long running times

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

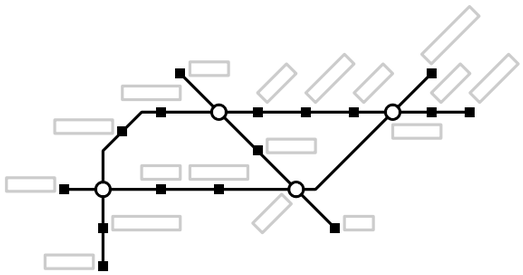
Visualization of Graphs



Lecture 12:
Octilinear Graph Drawing
Metro Map Layout



Part V:
Mixed-Integer Programming



Alexander Wolff

Mixed-Integer Programming

[Nöllenburg & Wolff 2011]

- find exact optimum solution using combinatorial optimization

Mixed-Integer Programming

[Nöllenburg & Wolff 2011]

- find exact optimum solution using combinatorial optimization
- split design rules into **hard** and **soft** constraints

Mixed-Integer Programming

[Nöllenburg & Wolff 2011]

- find exact optimum solution using combinatorial optimization
- split design rules into **hard** and **soft** constraints
- model constraints as linear (in)equalities and linear objective function
  $\rightarrow$ mixed-integer programming **MIP**

Mixed-Integer Programming

[Nöllenburg & Wolff 2011]

- find exact optimum solution using combinatorial optimization
- split design rules into **hard** and **soft** constraints
- model constraints as linear (in)equalities and linear objective function
  $\rightarrow$ mixed-integer programming **MIP**
- integrate overlap-free station labeling in same model

Mixed-Integer Programming

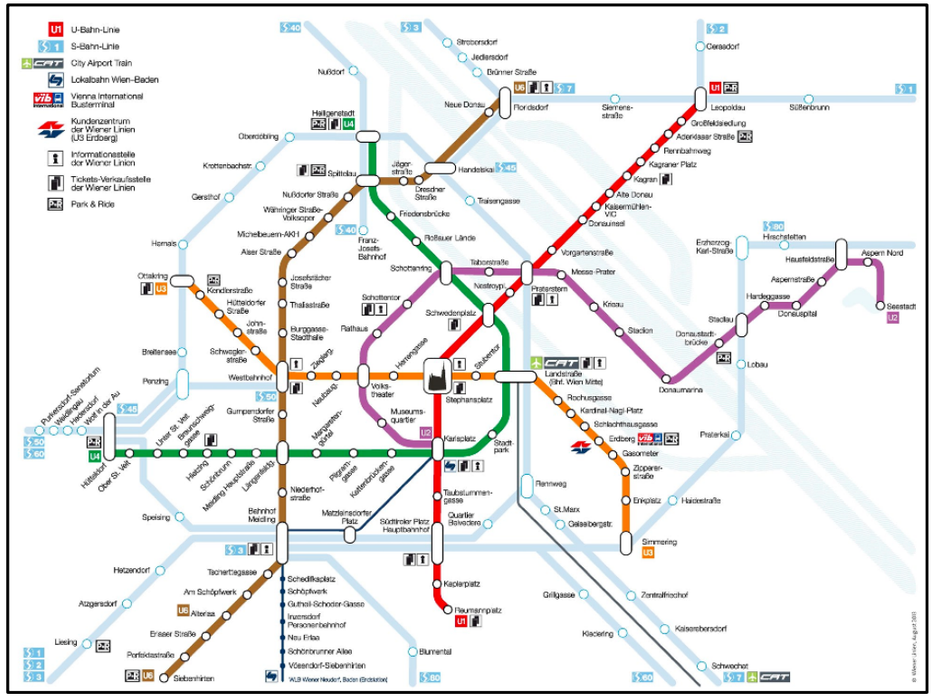
[Nöllenburg & Wolff 2011]

- find exact optimum solution using combinatorial optimization
- split design rules into **hard** and **soft** constraints
- model constraints as linear (in)equalities and linear objective function
  $\rightarrow$ mixed-integer programming **MIP**
- integrate overlap-free station labeling in same model
- can use sophisticated optimization tools as black box solvers
  (e.g., CPLEX, Gurobi)

Hard Constraints

(R1) preserve embedding/**topology** and **planarity**

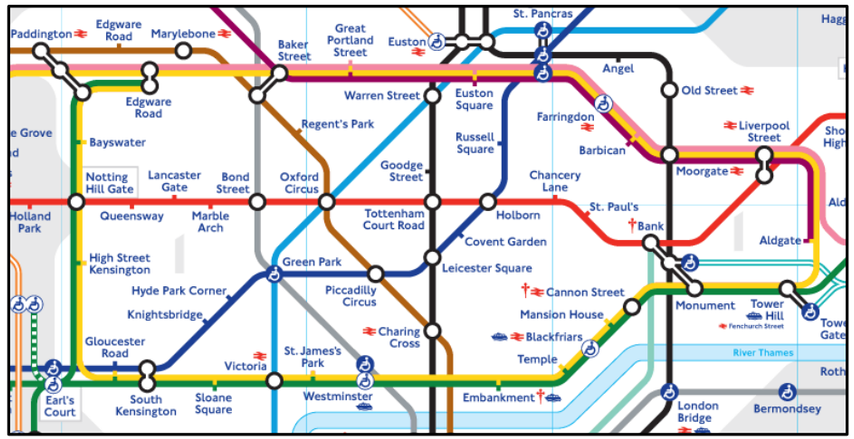


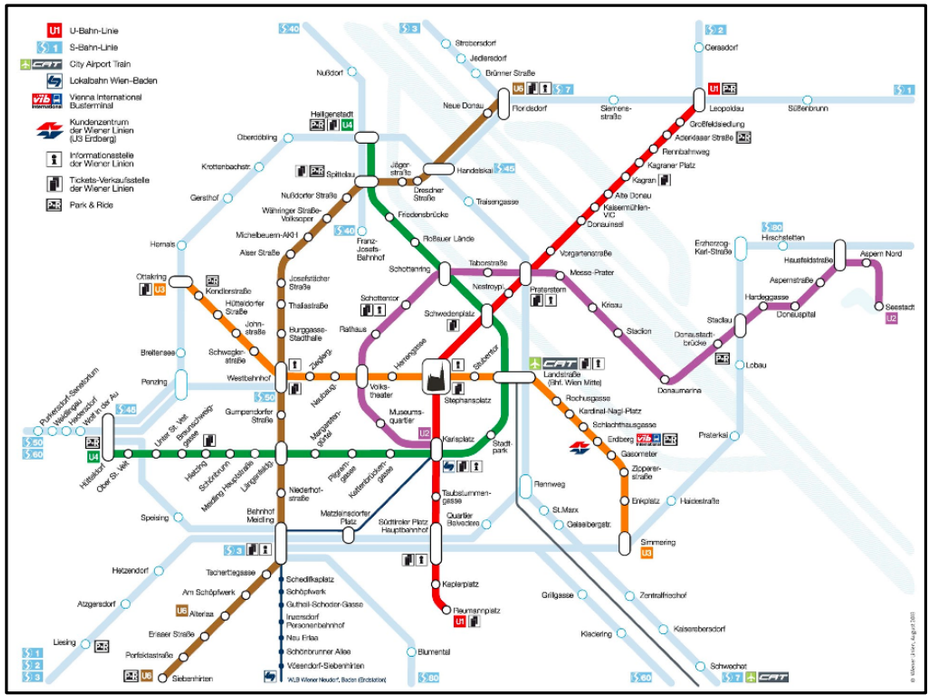

U-Bahn-Linie
S-Bahn-Linie
City Airport Train
Lokalbahn Wien–Baden
Vienna International Busterminal
Kundenzentrum der Wiener Linien (U3 Erdberg)
Informationsstelle der Wiener Linien
Tickets-Verkaufsstelle der Wiener Linien
Park & Ride
Stephansplatz
Karlsplatz
Schwedenplatz
Praterstern
Westbahnhof
Hauptbahnhof
Landstraße (Bhf. Wien Mitte)
Volkstheater
Museumsquartier
Rathaus
Schottentor
Schottenring
Taborstraße
Nestroyplatz
Messe-Prater
Krieau
Stadion
Donaumarina
Donauspital
Aspern Nord
Seestadt
Leopoldau
Floridsdorf
Heiligenstadt
Spittelau
Friedensbrücke
Roßauer Lände
Handelskai
Nußdorf
Heiligenstadt
Oberdöbling
Gersthof
Hernals
Alser Straße
Josefstädter Straße
Thaliastraße
Burggasse-Stadthalle
Schwarzenbergplatz
Zieglergasse
Herrengasse
Operngasse
Kärntner Ring/Kärntner Straße
Graben
Rotenturmstraße
Rochusgasse
Kardinal-Nagl-Platz
Schlachthausgasse
Prater Hauptallee
Lusthaus
Ausstellungsstraße
Messegelände
Engerthstraße
Traisengasse
Dresdner Straße
Jägerstraße
Nordwestbahnhof
Nordbahnhof
Praterstern
Taubstummengasse
Quartier Belvedere
Südtiroler Platz Hauptbahnhof
Matzleinsdorfer Platz
Margaretengürtel
Pilgramgasse
Reinprechtsdorfer Straße
Längenfeldgasse
Meidlinger Hauptstraße
Niederhofstraße
Schönbrunn
Hietzing
Unter St. Veit
Braunhubergasse
Gumpendorfer Straße
Neubaugasse
Zollergasse
Perolting
Breitenfeldergasse
Schwarzenbergstraße
Johnstraße
Hütteldorfer Straße
Kliebergasse
Josefstädter Straße
Otto-Wagner-Platz
Währinger Straße-Volksoper
Michelbeuern-AKH
Franz-Josefs-Bahnhof
Tabor
Vorgartenstraße
Erzherzog-Karl-Straße
Hirschstetten
Hausfeldstraße
Aspernstraße
Hardeggasse
Stadlau
Donaustadtbrücke
Lobau
Kagraner Platz
Kagran
Alte Donau
Kaiserin-Elisabeth-Westbahn
Donauinsel
Großfeldsiedlung
Adolf-Loos-Straße
Rennbahnweg
Siemens-straße
Süßenbrunn
Strebersdorf
Jedlersdorf
Brünner Straße
Neue Donau
Floridsdorf
Ceresdorf
Gasometer
Zippererstraße
Enkplatz
Haidestraße
Simmering
Kaiserebersdorf
Schwechat
Zentralfriedhof
Kledering
Grillgasse
Rennweg
St.Marx
Gaisbergstr.
Kaiserstraße
Kepplerplatz
Neumannplatz
Blumental
Siebenhirten
Liesing
Perfektastraße
Eraser Straße
Alterlaa
Atzgersdorf
Hetzendorf
Tscherttegasse
Schedifkaplatz
Schöpfwerk
Am Schöpfwerk
Gutheil-Schoder-Gasse
Inzersdorf Personenbahnhof
Neu Erlaa
Schönbrunner Allee
Vösendorf-Siebenhirten
Speising
Bahnhof Meidling
Niederhofstraße
Hütteldorf
Ober St. Veit
Unter St. Veit
Braunhubergasse
Hietzing
Schönbrunn
Meidling Hauptstraße
Längenfeldgasse
Pilgramgasse
Reinprechtsdorfer Straße
Margaretengürtel
Pötzleinsdorf
Währing
Weinhaus
Hernals
Gersthof
Oberdöbling
Krottenbachstr.
Nußdorf
Heiligenstadt
Spittelau
Friedensbrücke
Roßauer Lände
Handelskai

Hard Constraints

(R1) preserve embedding/**topology** and **planarity**

(R2) draw all edges **octilinearly**



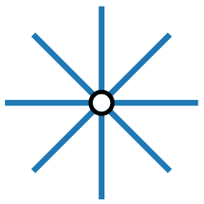


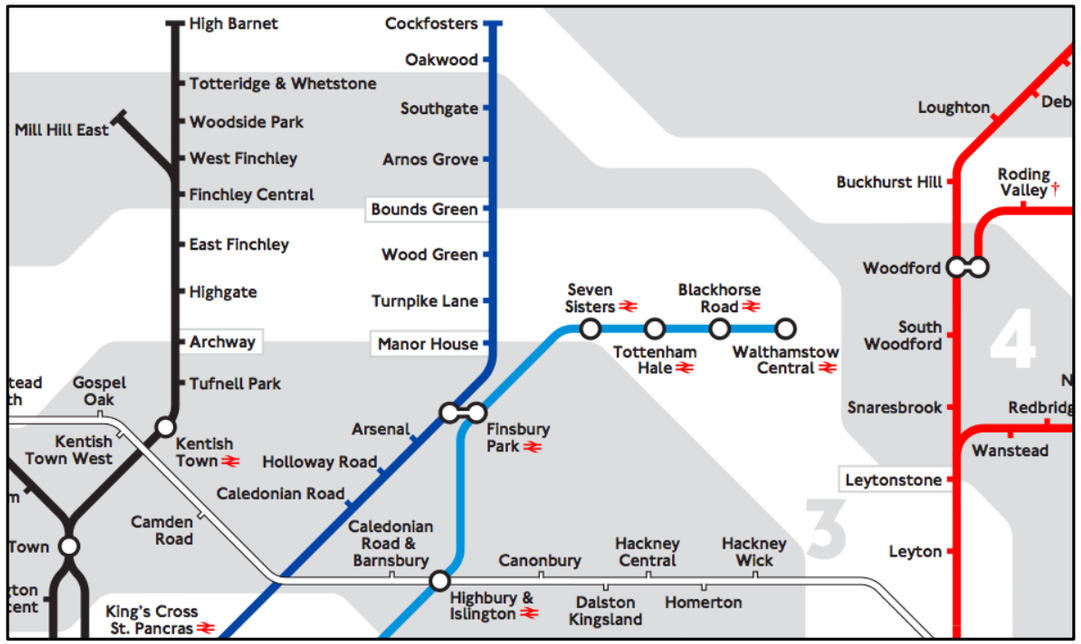
Hard Constraints

(R1) preserve embedding/**topology** and **planarity**

(R2) draw all edges **octilinearly**

(R7) enforce **minimum edge length** $\ell_{\min}$





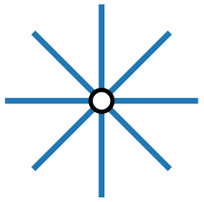
Hard Constraints

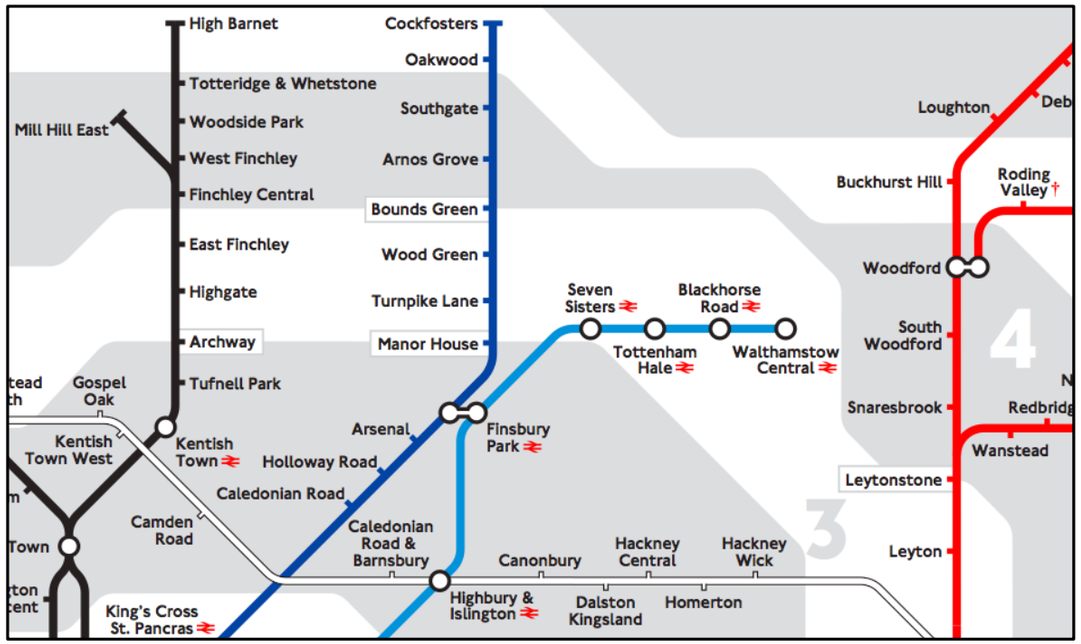
(R1) preserve embedding/**topology** and **planarity**

(R2) draw all edges **octilinearly**

(R7) enforce **minimum edge length** ℓ_{min}

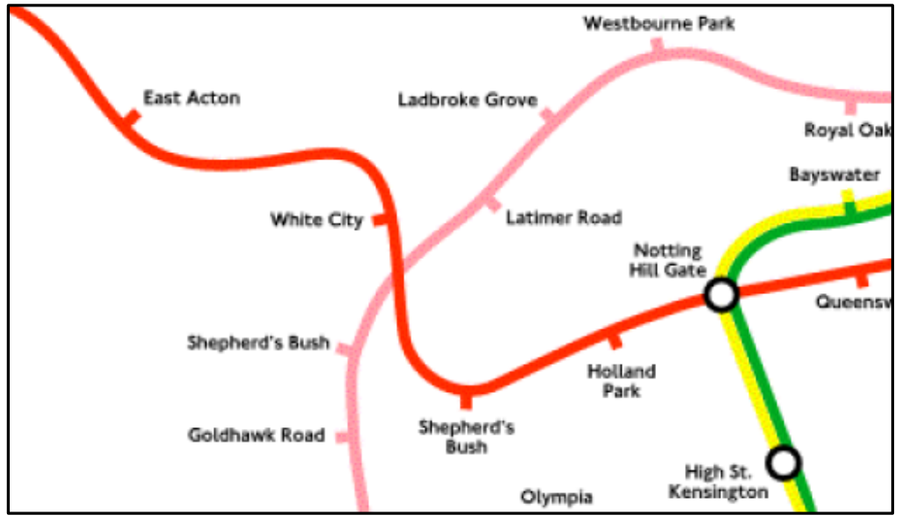
(R8) enforce **minimum feature separation** d_{min}

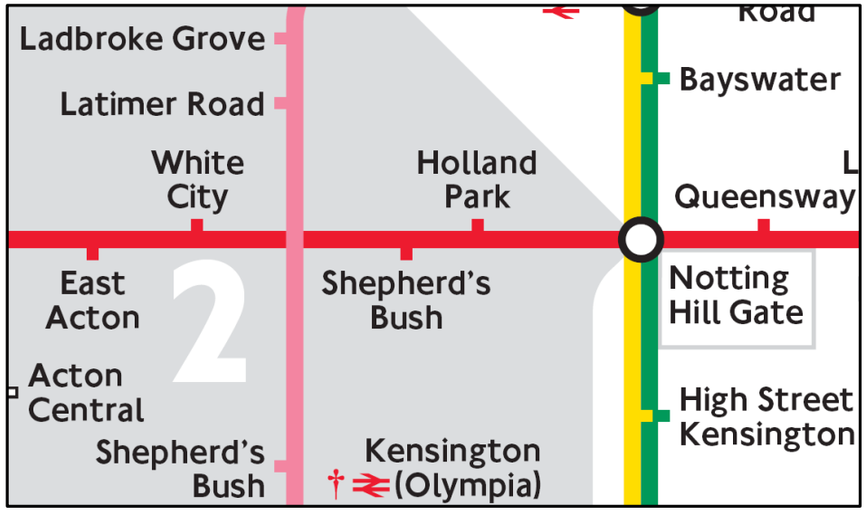




Soft Constraints

(R3+R4) draw lines in $\mathcal{L}$ with **few bends**



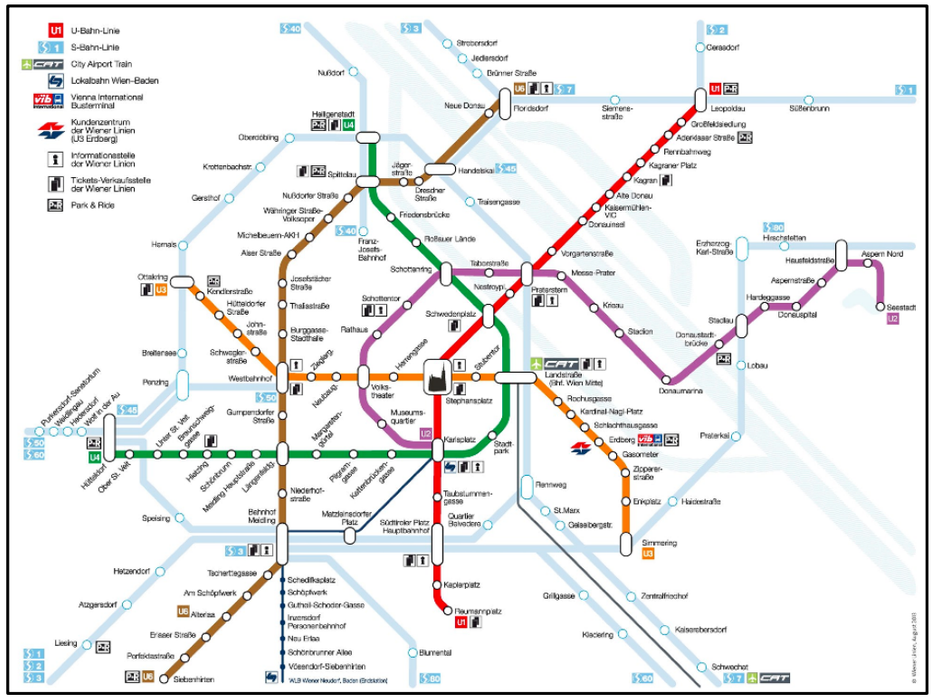


Soft Constraints

(R3+R4) draw lines in $\mathcal{L}$ with **few bends**

(R6) minimize geometric **distortion**



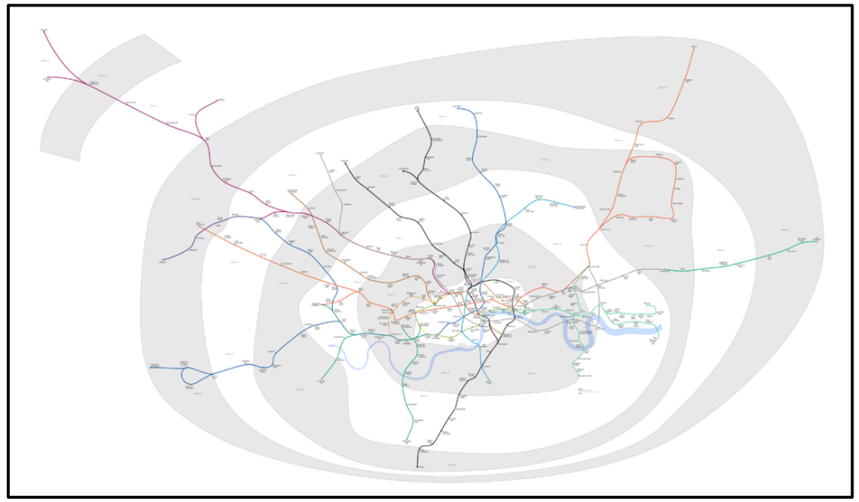

U-Bahn-Linie
S-Bahn-Linie
City Airport Train
Lokalbahn Wien–Baden
Vienna International Busterminal
Kundenzentrum der Wiener Linien (U3 Erdberg)
Informationsstelle der Wiener Linien
Tickets-Verkaufsstelle der Wiener Linien
Park & Ride
Stephansplatz
Karlsplatz
Schwedenplatz
Westbahnhof
Praterstern
Hauptbahnhof
Landstraße (Bhf. Wien Mitte)
Volkstheater
Museumsquartier
Rathaus
Schottentor
Schottenring
Taborstraße
Nestroyplatz
Messe-Prater
Krieau
Stadion
Donaumarina
Donaustadtbrücke
Leopoldau
Floridsdorf
Heiligenstadt
Spittelau
Handelskai
Friedensbrücke
Rossauer Lände
Franz-Josefs-Bahnhof
Meidling Hauptstraße
Matzleinsdorfer Platz
Südtiroler Platz Hauptbahnhof
Quartier Belvedere
Reumannplatz
Kaiserebersdorf
Simmering
Zentralfriedhof
Aspern Nord
Seestadt
Hausfeldstraße
Aspernstraße
Hardeggasse
Donauspital
Erzherzog-Karl-Straße
Hirschstetten
Großfeldsiedlung
Aderklaaser Straße
Rennbahnweg
Kagraner Platz
Kagran
Alte Donau
Kaisermühlen-VIC
Donauinsel
Süßenbrunn
Strebersdorf
Jedlersdorf
Brünner Straße
Nußdorf
Neue Donau
Siemens-straße
Cerasdorf
Oberdöbling
Krottenbachstr.
Gersthof
Nußdorfer Straße
Währinger Straße-Volksoper
Michelbeuern-AKH
Alser Straße
Hernals
Josefstädter Straße
Thaliastraße
Hütteldorfer Straße
Kendlerstraße
Ottakring
Jörgerstraße
Dresdner Straße
Traisengasse
Burggasse-Stadthalle
Schwarzenbergplatz
Lothringerstraße
Johannesgasse
Zieglergasse
Neubaugasse
Mariahilfer Straße
Babenbergerstraße
Operngasse
Kärntnerring
Kärntnerstraße
Graben
Rotenturmstraße
Herrengasse
Schauflergasse
Kohlmarkt
Michaelerplatz
Am Hof
Freyung
Schottenring
Universitätsstraße
Rooseveltplatz
Schottentor
Währinger Straße
Dr.-Karl-Lueger-Platz
Rathaus
Volkstheater
Museumsquartier
Karlsplatz
Stadtpark
Landstraße
Wien Mitte
Rochusgasse
Kardinal-Nagl-Platz
Schlachthausgasse
Erdberg
Gasometer
Zippererstraße
Enkplatz
Haidestraße
Prater
Prater Hauptallee
Praterstern
Messe-Prater
Vorgartenstraße
Taborstraße
Nestroyplatz
Schwedenplatz
Stadion
Krieau
Donaumarina
Donaustadtbrücke
Stadlau
Lobau
Donauspital
Hardeggasse
Aspernstraße
Hausfeldstraße
Aspern Nord
Seestadt
Hirschstetten
Erzherzog-Karl-Straße
Großfeldsiedlung
Aderklaaser Straße
Rennbahnweg
Kagraner Platz
Kagran
Alte Donau
Kaisermühlen-VIC
Donauinsel
Leopoldau
Floridsdorf
Süßenbrunn
Cerasdorf
Strebersdorf
Jedlersdorf
Brünner Straße
Nußdorf
Neue Donau
Siemens-straße
Heiligenstadt
Oberdöbling
Krottenbachstr.
Spittelau
Gersthof
Nußdorfer Straße
Währinger Straße-Volksoper
Michelbeuern-AKH
Alser Straße
Hernals
Josefstädter Straße
Thaliastraße
Hütteldorfer Straße
Kendlerstraße
Ottakring
Jörgerstraße
Dresdner Straße
Handelskai
Traisengasse
Friedensbrücke
Rossauer Lände
Franz-Josefs-Bahnhof
Schottenring
Taborstraße
Nestroyplatz
Messe-Prater
Praterstern
Vorgartenstraße
Krieau
Stadion

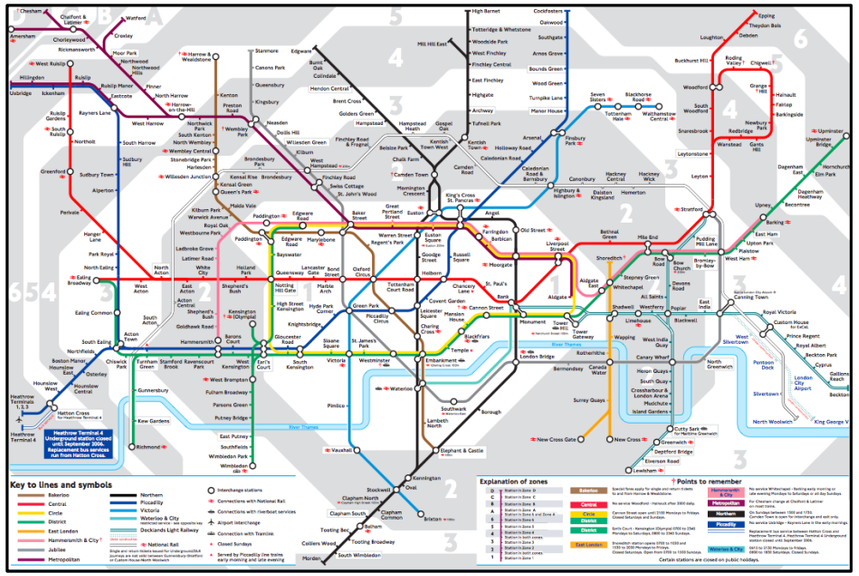
Soft Constraints

(R3+R4) draw lines in $\mathcal{L}$ with **few bends**

(R6) minimize geometric **distortion**

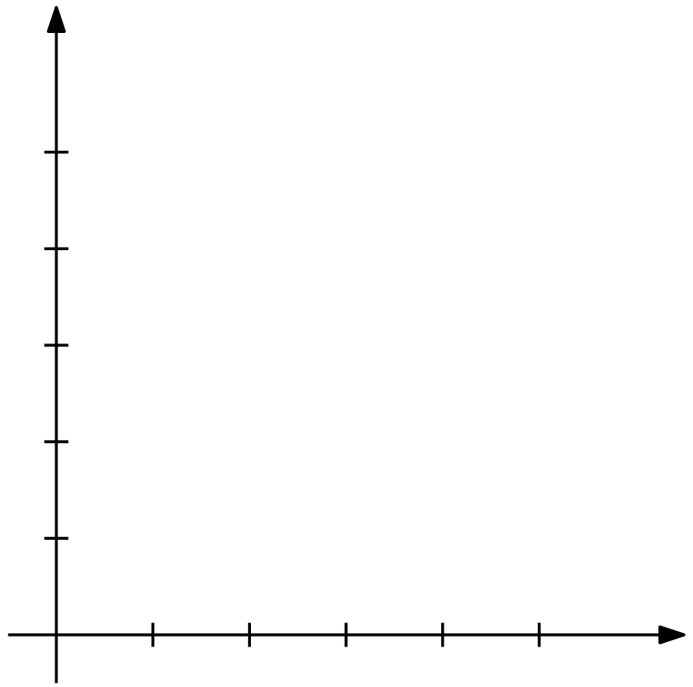
(R7) minimize total **edge length**





Linear Programming

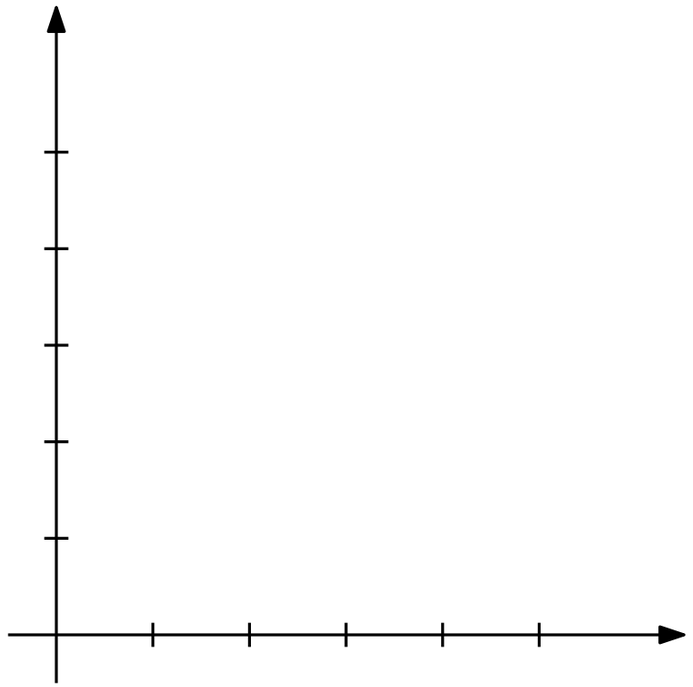
Linear Programming (LP) is an efficient optimization method for



Linear Programming

Linear Programming (LP) is an efficient optimization method for

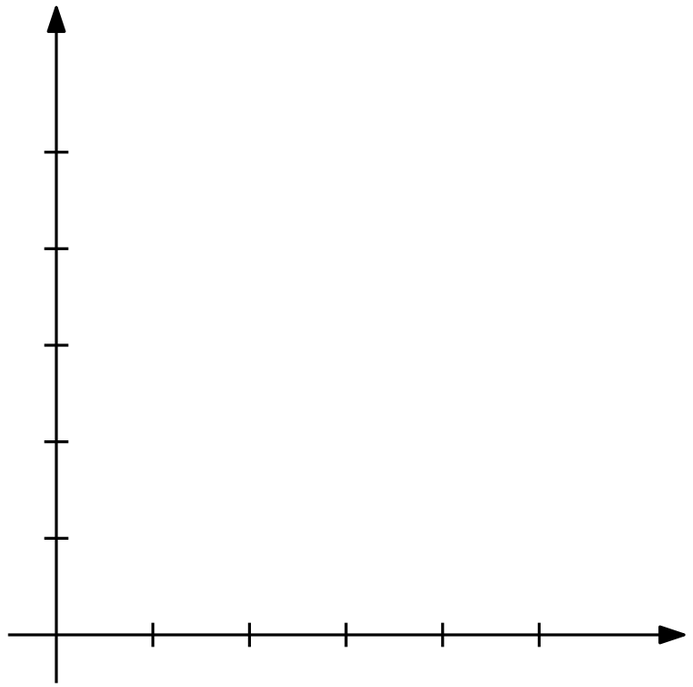
- linear constraints



Linear Programming

Linear Programming (LP) is an efficient optimization method for

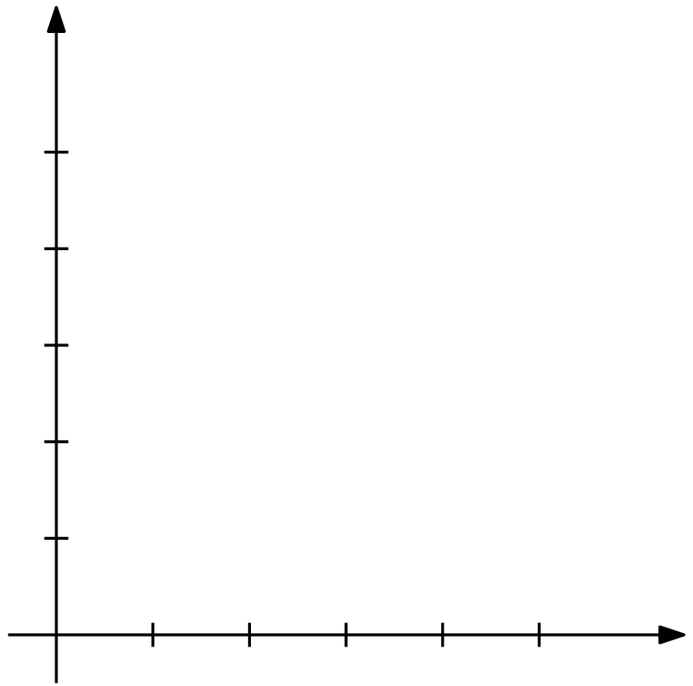
- linear constraints
- linear objective function



Linear Programming

Linear Programming (LP) is an efficient optimization method for

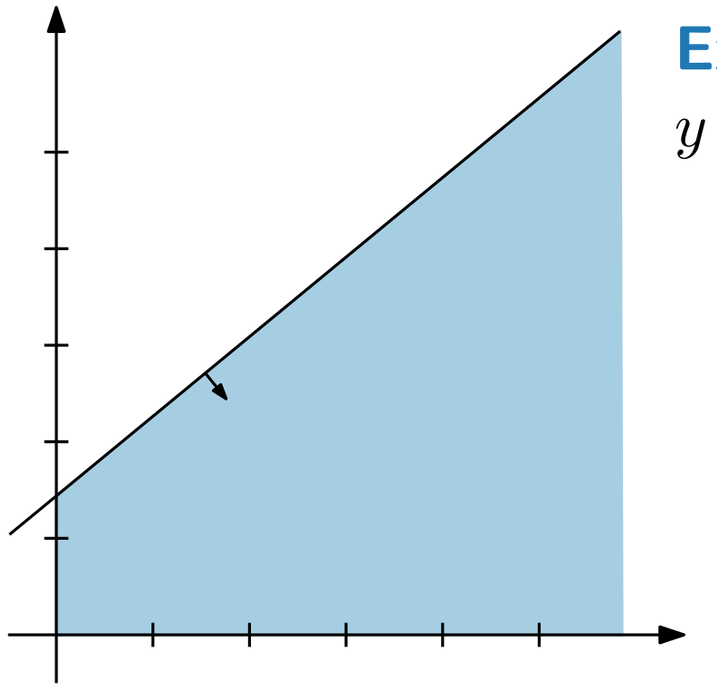
- linear constraints
- linear objective function
- real-valued variables



Linear Programming

Linear Programming (LP) is an efficient optimization method for

- linear constraints
- linear objective function
- real-valued variables



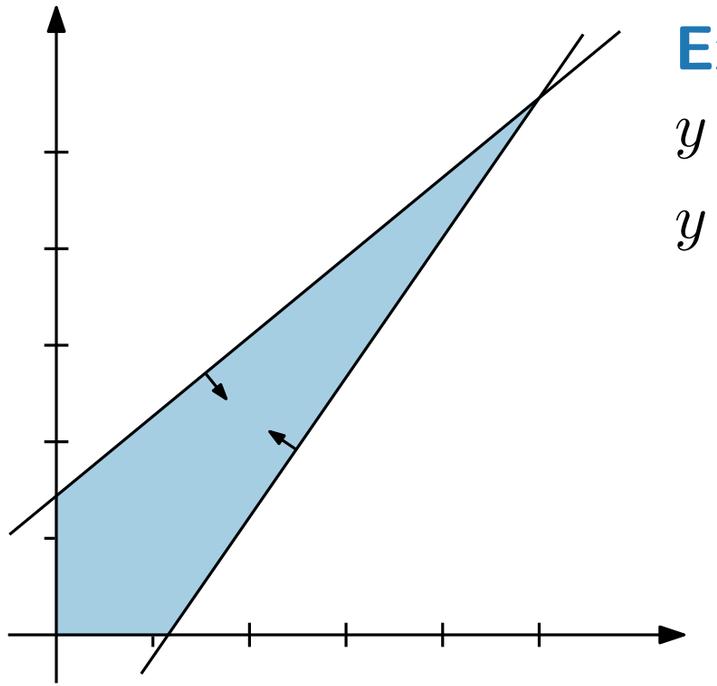
Example.

$y \leq 0.9x + 1.5$

Linear Programming

Linear Programming (LP) is an efficient optimization method for

- linear constraints
- linear objective function
- real-valued variables



Example.

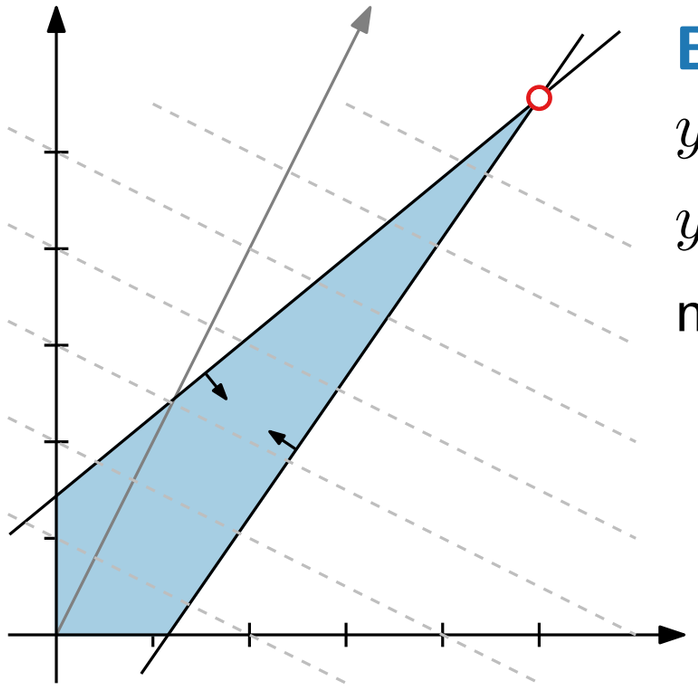
$y \leq 0.9x + 1.5$

$y \geq 1.4x - 1.3$

Linear Programming

Linear Programming (LP) is an efficient optimization method for

- linear constraints
- linear objective function
- real-valued variables



Example.

$y \leq 0.9x + 1.5$

$y \geq 1.4x - 1.3$

maximize $x + 2y$

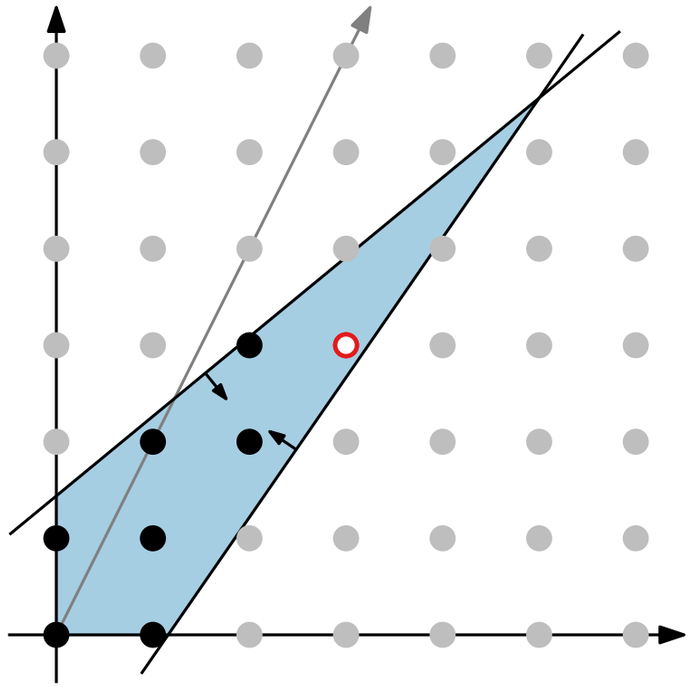
Linear Programming

Linear Programming (LP) is an efficient optimization method for

- linear constraints
- linear objective function
- real-valued variables

Mixed Integer Programming (MIP)

- in addition binary and integer variables



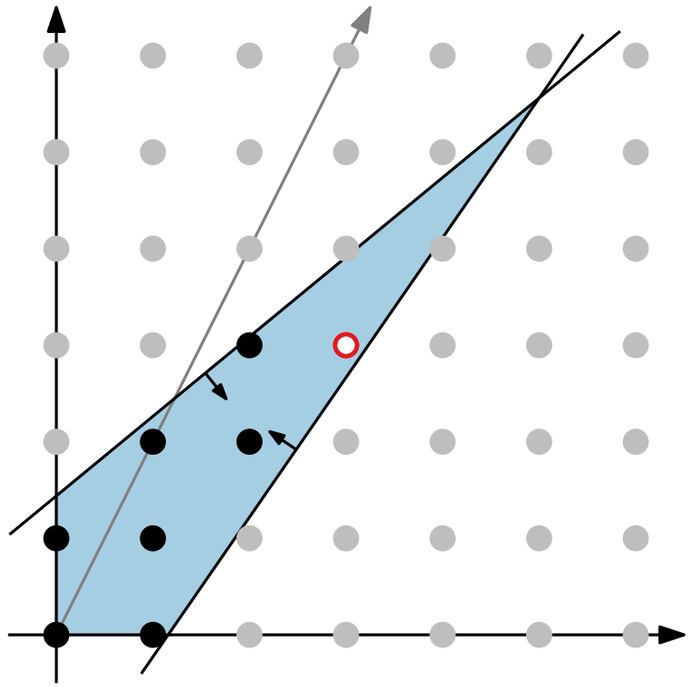
Linear Programming

Linear Programming (LP) is an efficient optimization method for

- linear constraints
- linear objective function
- real-valued variables

Mixed Integer Programming (MIP)

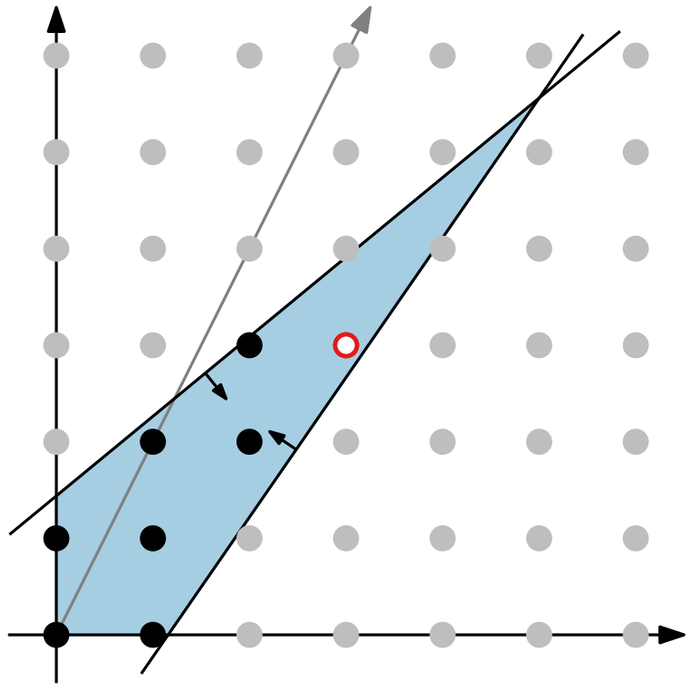
- in addition binary and integer variables
- NP-hard optimization problem



Linear Programming

Linear Programming (LP) is an efficient optimization method for

- linear constraints
- linear objective function
- real-valued variables



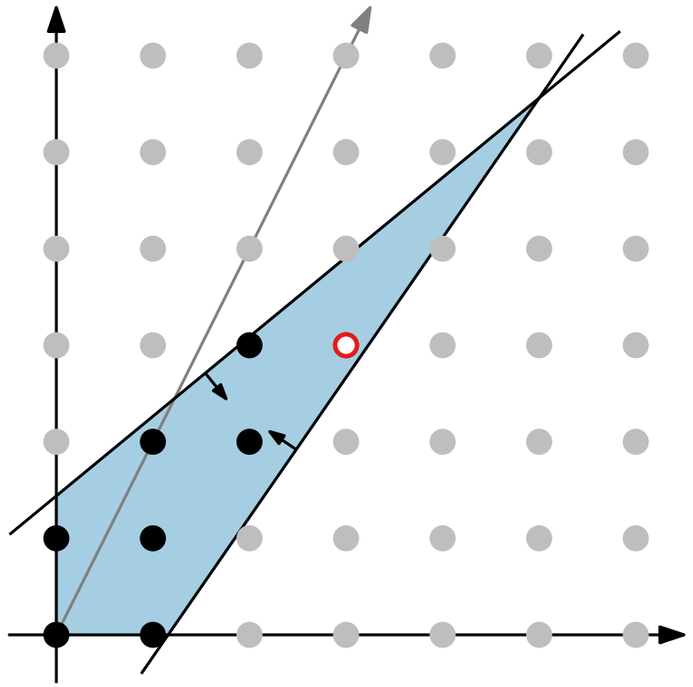
Mixed Integer Programming (MIP)

- in addition binary and integer variables
- NP-hard optimization problem
- still method of choice for many practical optimization tasks

Linear Programming

Linear Programming (LP) is an efficient optimization method for

- linear constraints
- linear objective function
- real-valued variables



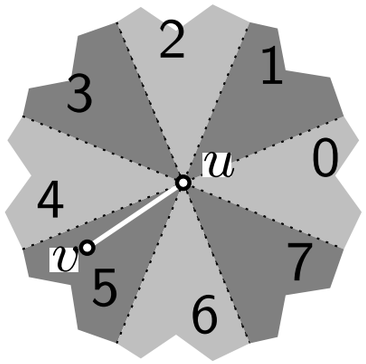
Mixed Integer Programming (MIP)

- in addition binary and integer variables
- NP-hard optimization problem
- still method of choice for many practical optimization tasks

Theorem 5.
Metro map layout can be modeled as MIP such that

- hard constraints $\rightarrow$ linear constraints
- soft constraints $\rightarrow$ linear objective function

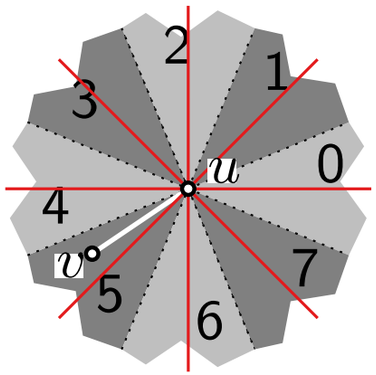
Sectors and Coordinates



Sectors.

- for each vertex u partition the plane into eight sectors numbered 0–7
  here: $\text{sec}(u, v) = 5$ in the input

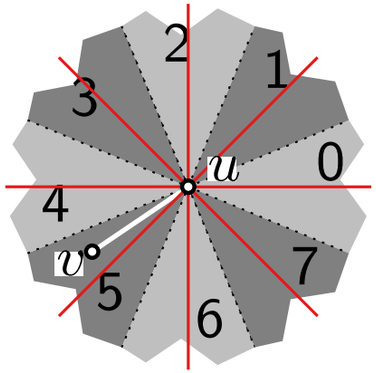
Sectors and Coordinates



Sectors.

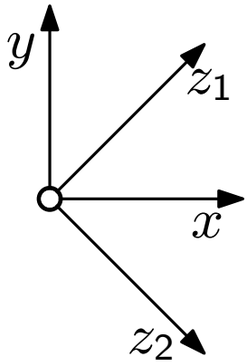
- for each vertex u partition the plane into eight sectors numbered 0–7
  here: $\text{sec}(u, v) = 5$ in the input
- number octilinear edge directions accordingly
  here, e.g., $\text{dir}(u, v) = 5$ in the solution (output)

Sectors and Coordinates



Sectors.

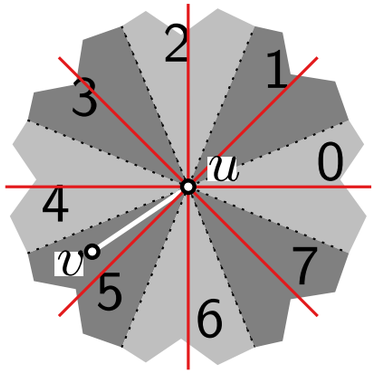
- for each vertex u partition the plane into eight sectors numbered 0–7
  here: $\text{sec}(u, v) = 5$ in the input
- number octilinear edge directions accordingly
  here, e.g., $\text{dir}(u, v) = 5$ in the solution (output)



Coordinates.

- assign (redundant) z_1- and z_2-coordinates to each vertex v
  - $z_1(v) = \frac{1}{2} \cdot (x(v) + y(v))$
  - $z_2(v) = \frac{1}{2} \cdot (x(v) - y(v))$

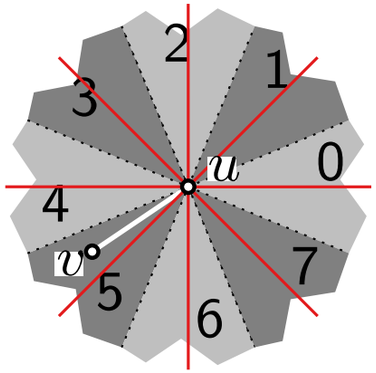
Octilinearity and Relative Position



Goal.

Draw the edge uv

Octilinearity and Relative Position

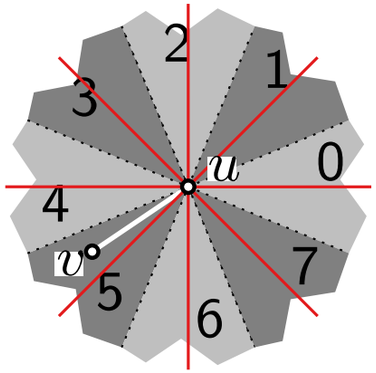


Goal.

Draw the edge uv

- octilinearly **(R2)**

Octilinearity and Relative Position

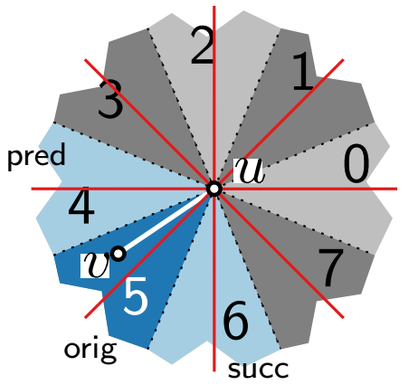


Goal.

Draw the edge uv

- octilinearly **(R2)**
- with minimum length $\ell = \ell_{uv}$ **(R7)**

Octilinearity and Relative Position

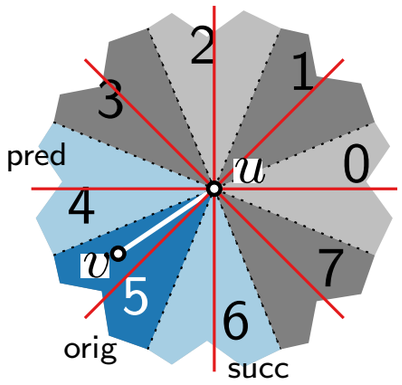


Goal.

Draw the edge uv

- octilinearly **(R2)**
- with minimum length $\ell = \ell_{uv}$ **(R7)**
- restricted to the three best directions **(R6)**

Octilinearity and Relative Position



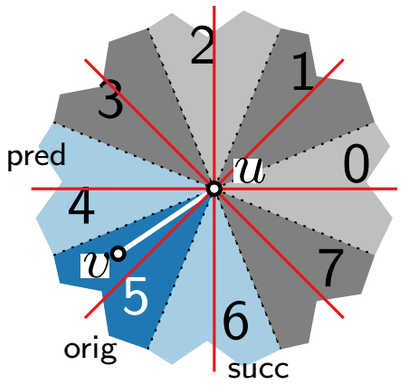
Goal.

Draw the edge uv

- octilinearly **(R2)**
- with minimum length $\ell = \ell_{uv}$ **(R7)**
- restricted to the three best directions **(R6)**

How to model this using linear constraints in a MIP?

Octilinearity and Relative Position



Goal.

Draw the edge uv

- octilinearly **(R2)**
- with minimum length $\ell = \ell_{uv}$ **(R7)**
- restricted to the three best directions **(R6)**

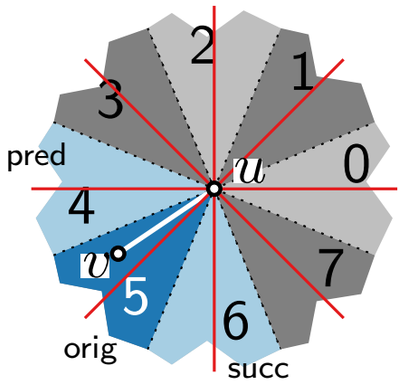
How to model this using linear constraints in a MIP?

Introduce **binary variables**

$$\alpha_{\text{pred}}(u, v) \quad \alpha_{\text{orig}}(u, v) \quad \alpha_{\text{succ}}(u, v)$$

to select exactly one of the three sectors.

Octilinearity and Relative Position



Goal.

Draw the edge uv

- octilinearly **(R2)**
- with minimum length $\ell = \ell_{uv}$ **(R7)**
- restricted to the three best directions **(R6)**

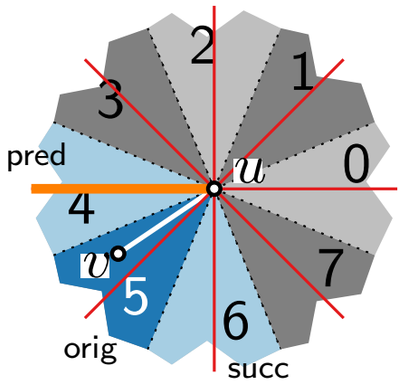
How to model this using linear constraints in a MIP?

Introduce **binary variables**

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

to select exactly one of the three sectors.

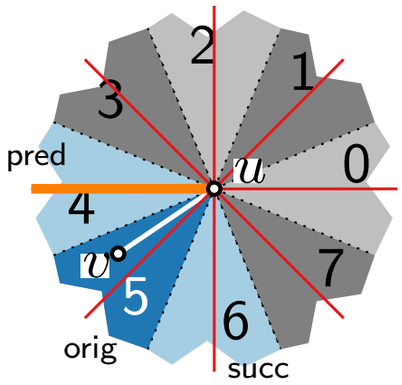
Octilinearity and Relative Position



Predecessor sector.

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

Octilinearity and Relative Position

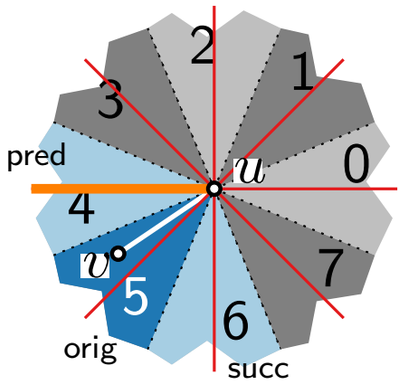


Predecessor sector.

very large constant

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

Octilinearity and Relative Position



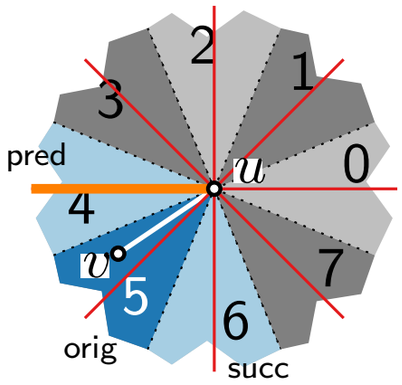
Predecessor sector.

very large constant

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

How does this work?

Octilinearity and Relative Position



Predecessor sector.

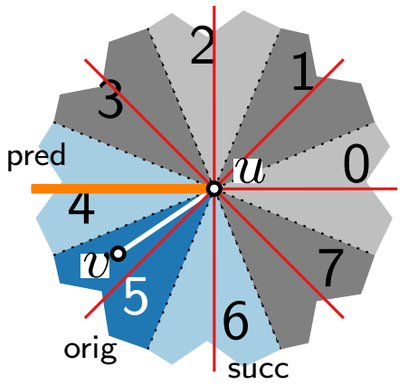
very large constant

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

How does this work?

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

Octilinearity and Relative Position



Predecessor sector.

very large constant

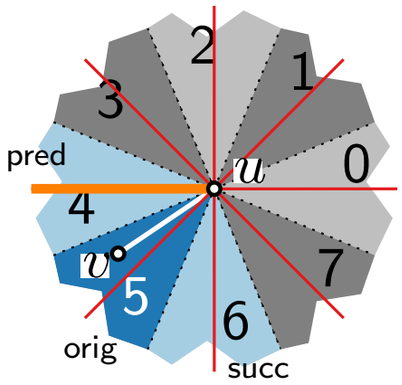
$$\begin{aligned} y(u) - y(v) &\leq M(1 - \alpha_{\text{pred}}(u,v)) \\ -y(u) + y(v) &\leq M(1 - \alpha_{\text{pred}}(u,v)) \\ x(u) - x(v) &\geq -M(1 - \alpha_{\text{pred}}(u,v)) + \ell \end{aligned}$$

How does this work?

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

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

Octilinearity and Relative Position



Predecessor sector.

very large constant

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

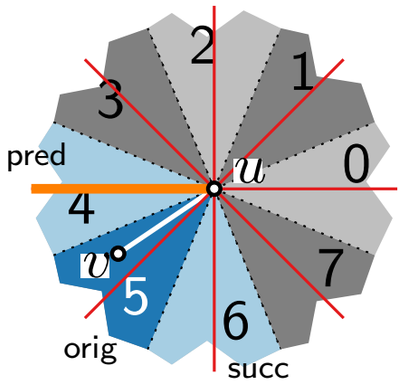
How does this work?

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

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

Case 2: $\alpha_{\text{pred}}(u,v) = 0$

Octilinearity and Relative Position



Predecessor sector.

very large constant

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

How does this work?

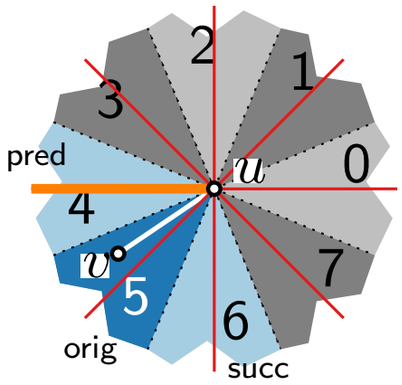
Case 1: $\alpha_{\text{pred}}(u,v) = 1$

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

Case 2: $\alpha_{\text{pred}}(u,v) = 0$

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

Octilinearity and Relative Position



Predecessor sector.

very large constant (how large, actually?)

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

How does this work?

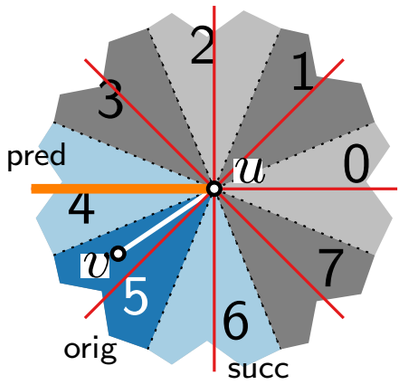
Case 1: $\alpha_{\text{pred}}(u,v) = 1$

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

Case 2: $\alpha_{\text{pred}}(u,v) = 0$

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

Octilinearity and Relative Position



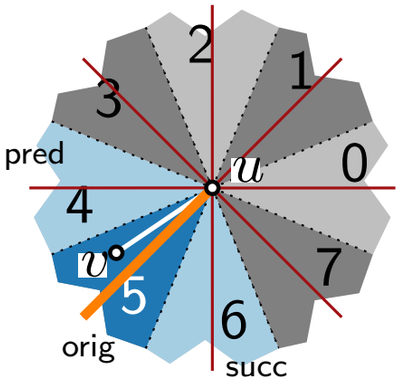
very large constant

Predecessor sector.

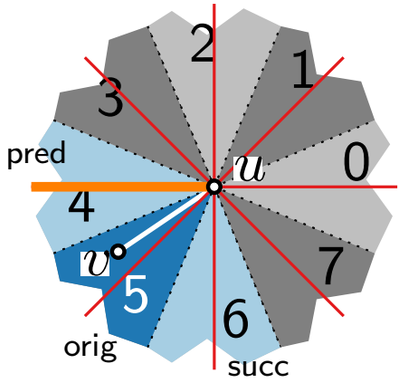
$$
\begin{aligned}
y(u) - y(v) &\leq M(1 - \alpha_{\mathsf{pred}}(u,v)) \\
-y(u) + y(v) &\leq M(1 - \alpha_{\mathsf{pred}}(u,v)) \\
x(u) - x(v) &\geq -M(1 - \alpha_{\mathsf{pred}}(u,v)) + \ell
\end{aligned}
$$

Original sector.

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



Octilinearity and Relative Position



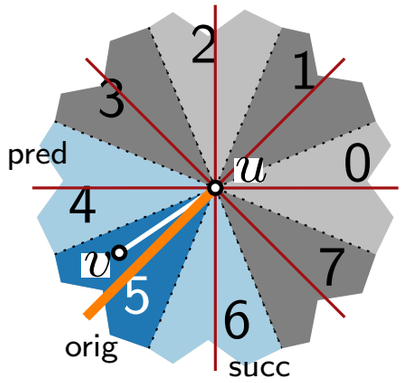
Predecessor sector.

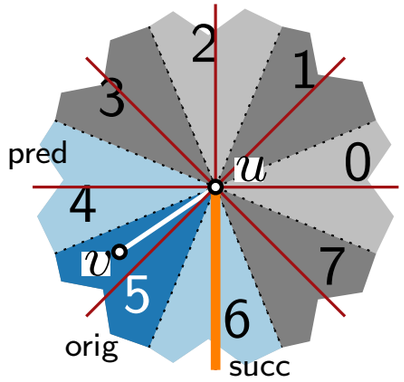
very large constant

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

Original sector.

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





Successor sector.

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

Objective Function

- models the three soft constraints

Objective Function

- models the three soft constraints
- weighted sum of individual cost functions

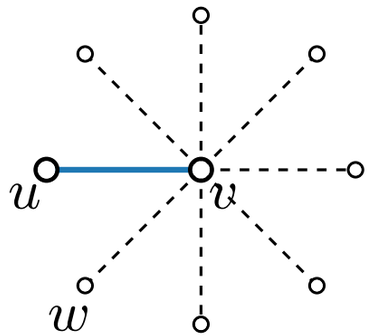
  minimize $\lambda_{\text{bends}}\,\text{cost}_{\text{bends}} + \lambda_{\text{length}}\,\text{cost}_{\text{length}} + \lambda_{\text{dist}}\text{cost}_{\text{dist}}$

Objective Function

- models the three soft constraints
- weighted sum of individual cost functions

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

Example: line bends (R3/R4)



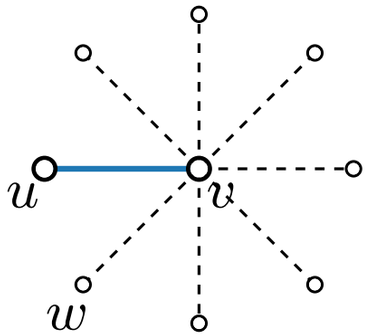
Objective Function

- models the three soft constraints
- weighted sum of individual cost functions

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

Example: line bends (R3/R4)

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

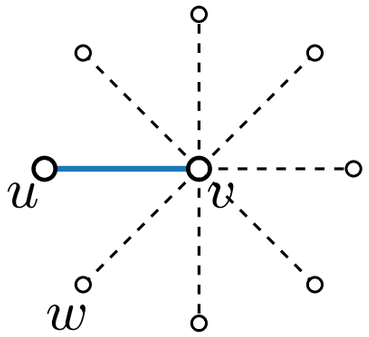


Objective Function

- models the three soft constraints
- weighted sum of individual cost functions

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

Example: line bends (R3/R4)



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

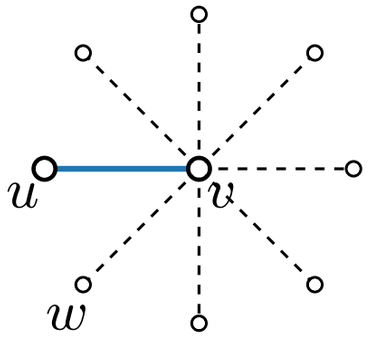
- draw as straight as possible

Objective Function

- models the three soft constraints
- weighted sum of individual cost functions

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

Example: line bends (R3/R4)



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

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

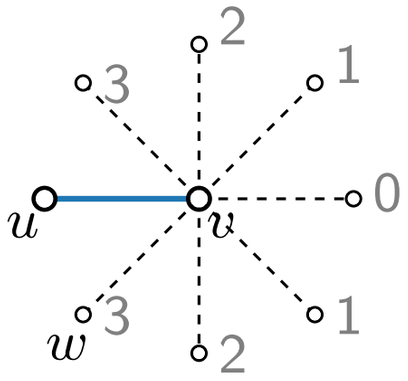
$$\text{cost}_{\text{bends}} = \sum_{L \in \mathcal{L}} \sum_{uv, vw \in L} \text{bend}(u, v, w)$$

Objective Function

- models the three soft constraints
- weighted sum of individual cost functions

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

Example: line bends (R3/R4)



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

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

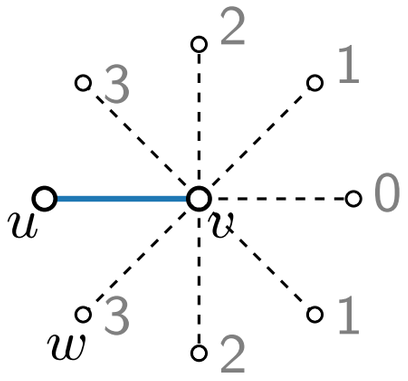
$$\text{cost}_{\text{bends}} = \sum_{L \in \mathcal{L}} \sum_{uv, vw \in L} \text{bend}(u, v, w)$$

Objective Function

- models the three soft constraints
- weighted sum of individual cost functions

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

Example: line bends (R3/R4)



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

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

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

To assign $\text{bend}(u, v, w)$ correctly, we need to define some linear constraints based on the direction variables $\text{dir}(u, v)$ and $\text{dir}(v, w)$.

Overview MIP model

Constraints.

- linearization of all hard constraints
- $O(n^2)$ variables and constraints (due to planarity)

Overview MIP model

Constraints.

- linearization of all hard constraints
- $O(n^2)$ variables and constraints (due to planarity)

Objective function.

- weighted sum of the three soft constraints
- minimize $\lambda_{\text{bend}}\text{cost}_{\text{bend}} + \lambda_{\text{len}}\text{cost}_{\text{len}} + \lambda_{\text{dist}}\text{cost}_{\text{dist}}$

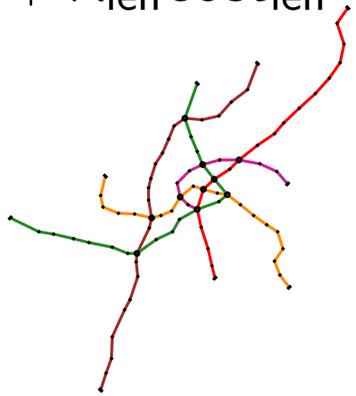
Overview MIP model

Constraints.

- linearization of all hard constraints
- $O(n^2)$ variables and constraints (due to planarity)

Objective function.

- weighted sum of the three soft constraints
- minimize $\lambda_{\text{bend}}\text{cost}_{\text{bend}} + \lambda_{\text{len}}\text{cost}_{\text{len}} + \lambda_{\text{dist}}\text{cost}_{\text{dist}}$
- effect:



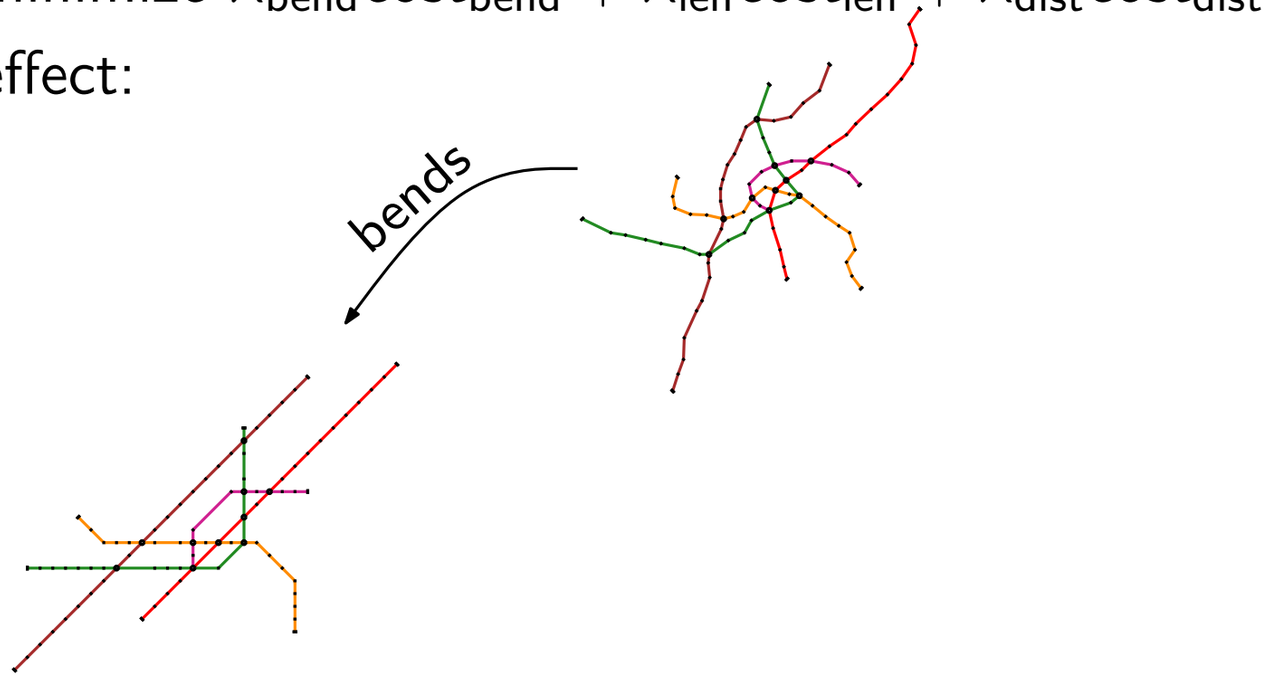
Overview MIP model

Constraints.

- linearization of all hard constraints
- $O(n^2)$ variables and constraints (due to planarity)

Objective function.

- weighted sum of the three soft constraints
- minimize $\lambda_{\text{bend}}\text{cost}_{\text{bend}} + \lambda_{\text{len}}\text{cost}_{\text{len}} + \lambda_{\text{dist}}\text{cost}_{\text{dist}}$
- effect:



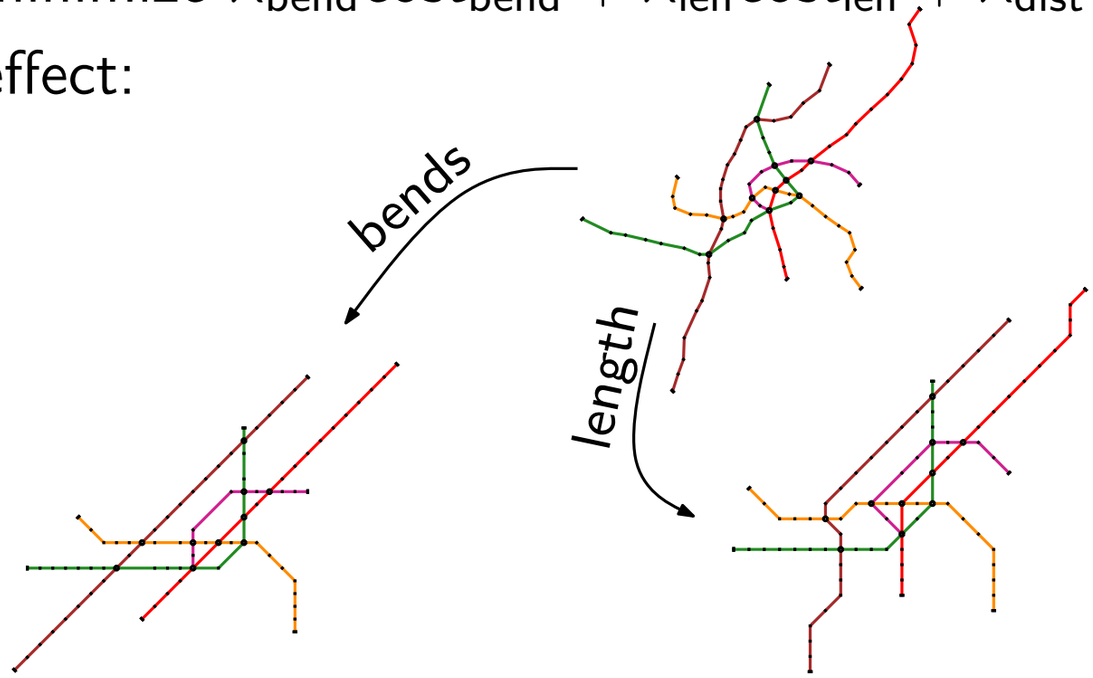
Overview MIP model

Constraints.

- linearization of all hard constraints
- $O(n^2)$ variables and constraints (due to planarity)

Objective function.

- weighted sum of the three soft constraints
- minimize $\lambda_{\text{bend}}\text{cost}_{\text{bend}} + \lambda_{\text{len}}\text{cost}_{\text{len}} + \lambda_{\text{dist}}\text{cost}_{\text{dist}}$
- effect:



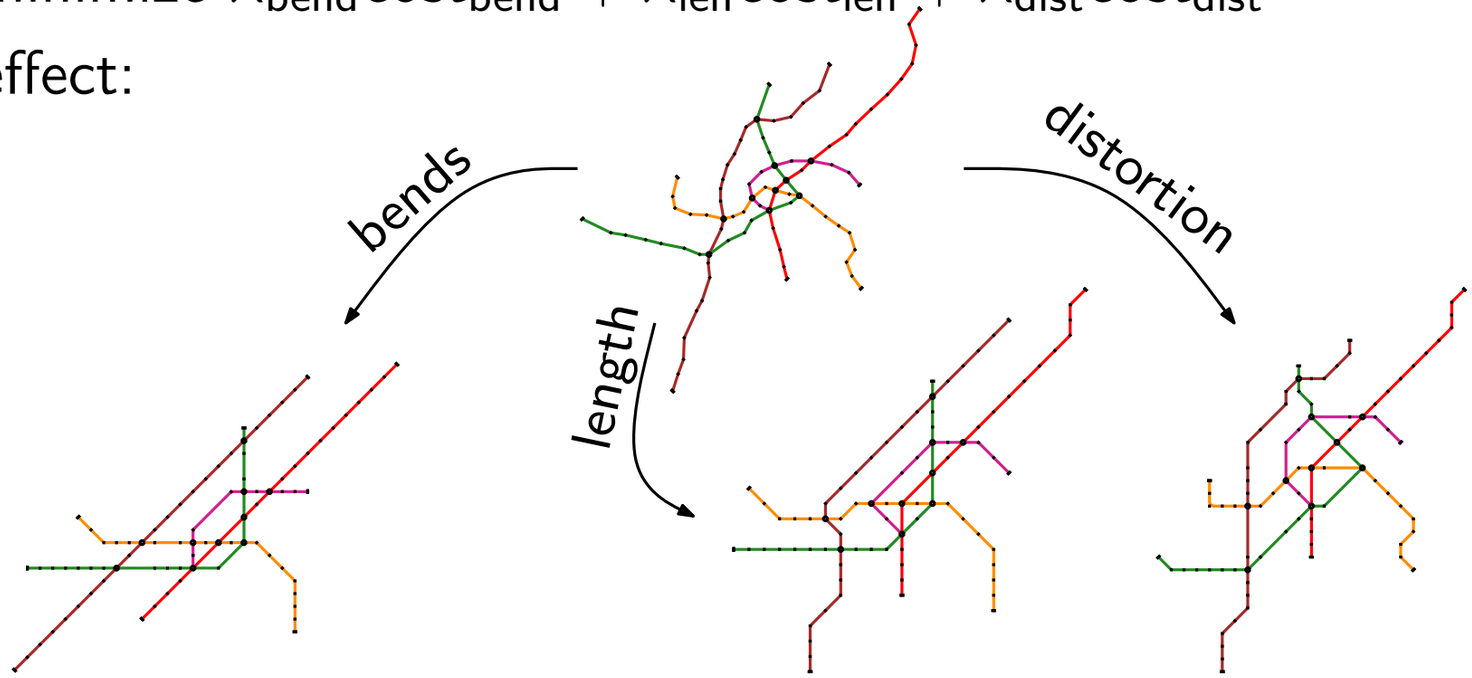
Overview MIP model

Constraints.

- linearization of all hard constraints
- $O(n^2)$ variables and constraints (due to planarity)

Objective function.

- weighted sum of the three soft constraints
- minimize $\lambda_{\text{bend}}\text{cost}_{\text{bend}} + \lambda_{\text{len}}\text{cost}_{\text{len}} + \lambda_{\text{dist}}\text{cost}_{\text{dist}}$
- effect:



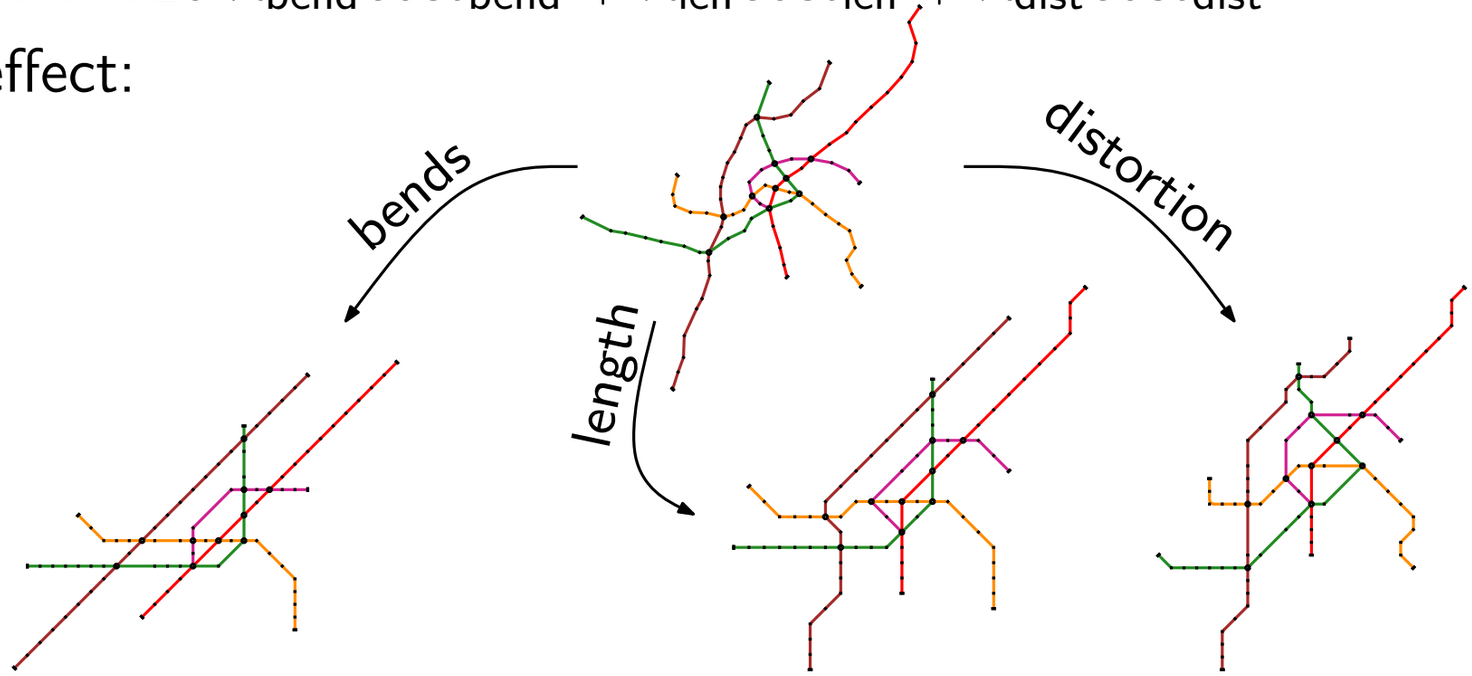
Overview MIP model

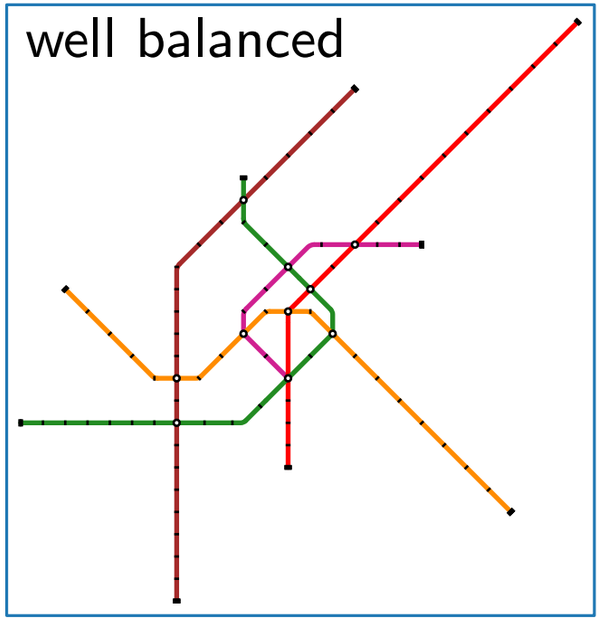
Constraints.

- linearization of all hard constraints
- $O(n^2)$ variables and constraints (due to planarity)

Objective function.

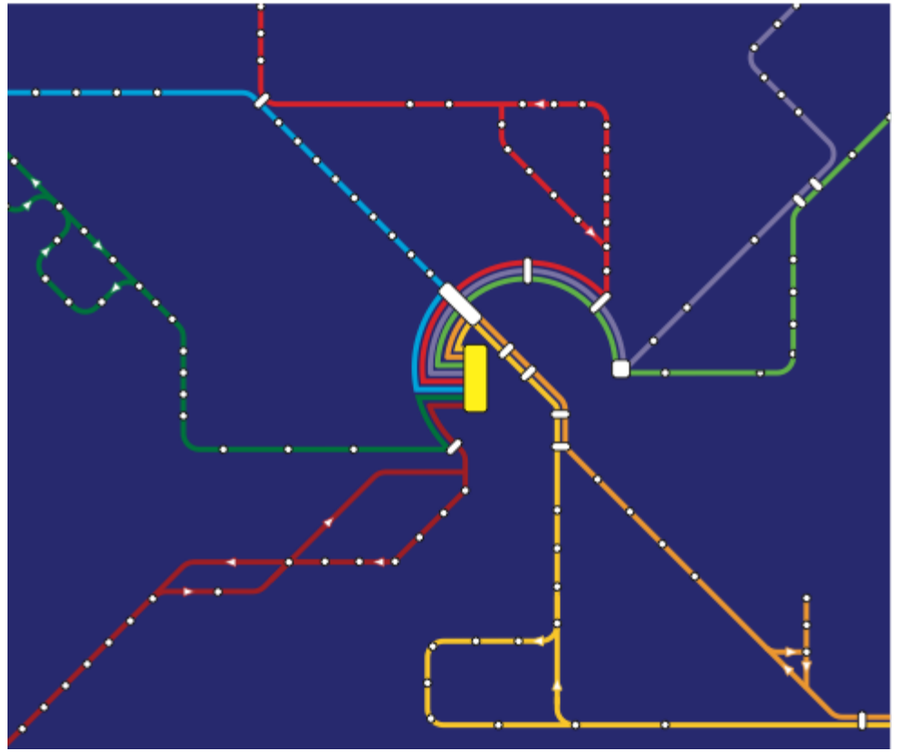
- weighted sum of the three soft constraints
- minimize $\lambda_{\text{bend}}\text{cost}_{\text{bend}} + \lambda_{\text{len}}\text{cost}_{\text{len}} + \lambda_{\text{dist}}\text{cost}_{\text{dist}}$
- effect:





Station Labeling

- unlabeled map mostly useless



Station Labeling

- unlabeled map mostly useless



Station Labeling

- unlabeled map mostly useless
- labels need space



Station Labeling

- unlabeled map mostly useless
- labels need space
- labels may not overlap each other



Station Labeling

- unlabeled map mostly useless
- labels need space
- labels may not overlap each other
- graph labeling problem is NP-hard
  [Kakoulis & Tollis 2001]

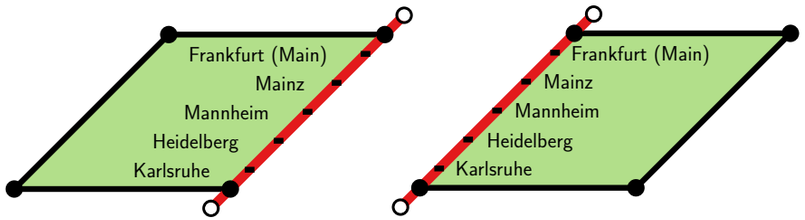


Station Labeling

- unlabeled map mostly useless
- labels need space
- labels may not overlap each other
- graph labeling problem is NP-hard
  [Kakoulis & Tollis 2001]

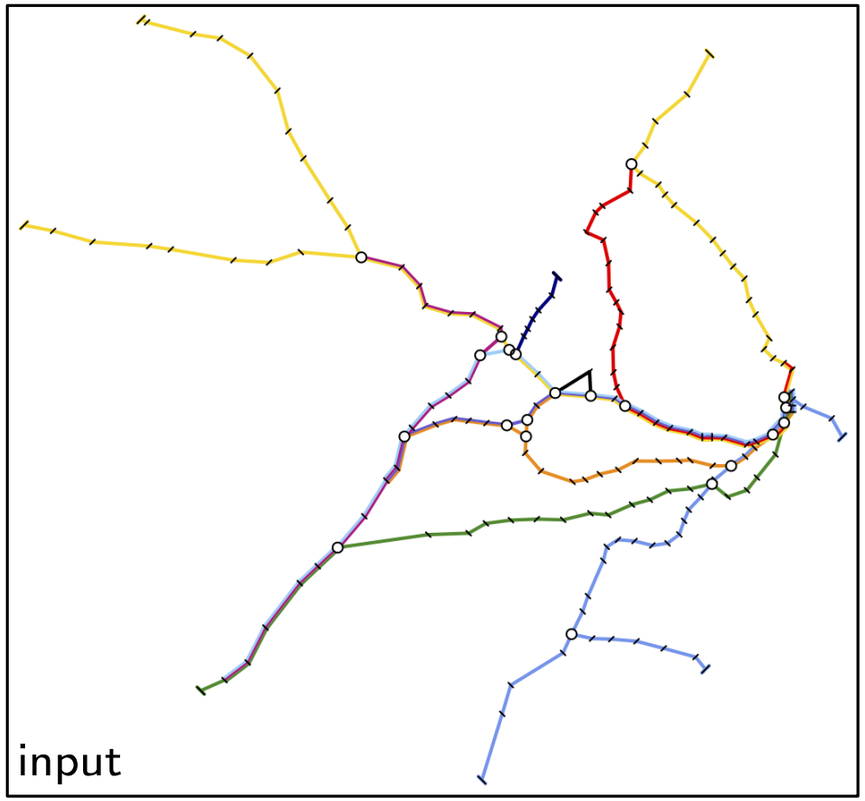


→ combine layout & labeling for optimal results!

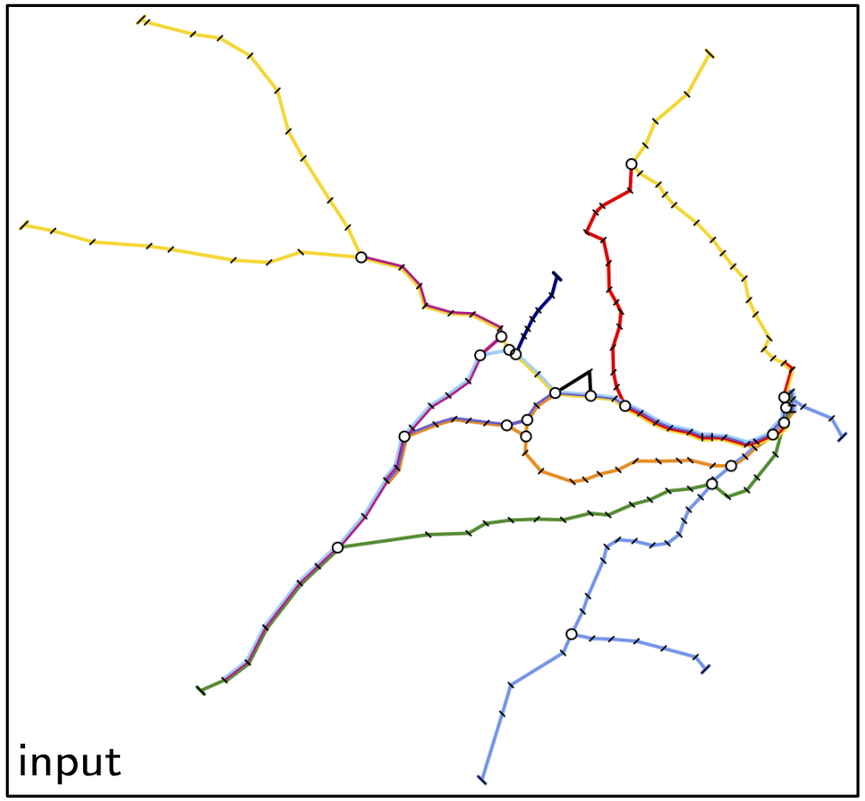


- parallelogram as special metro line
- switching sides allowed

Example: Sydney &

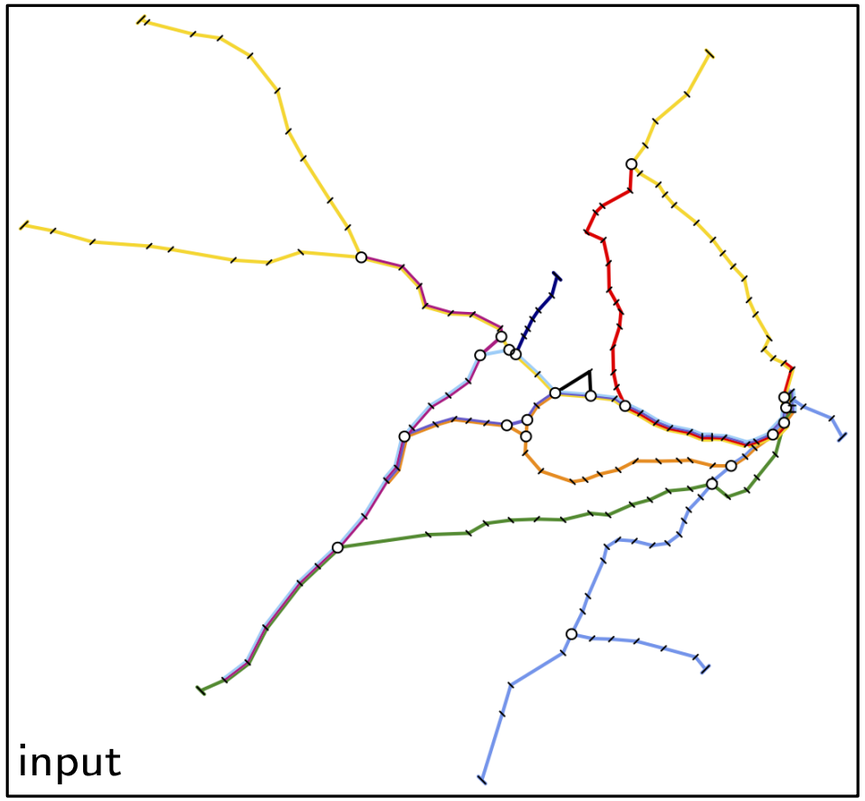


Example: Sydney &



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

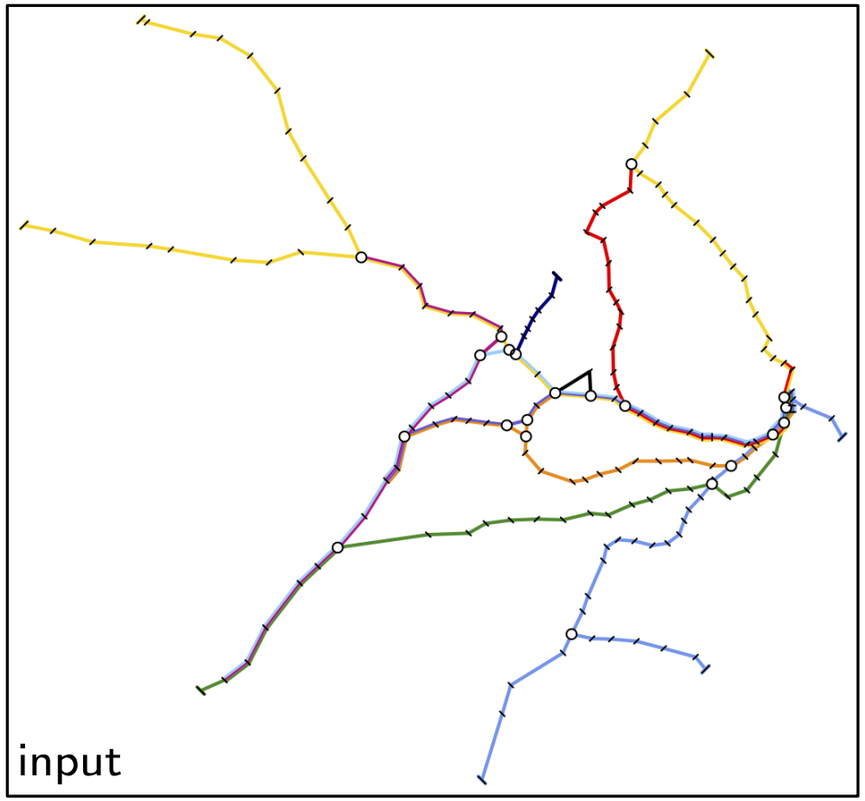
Example: Sydney &



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

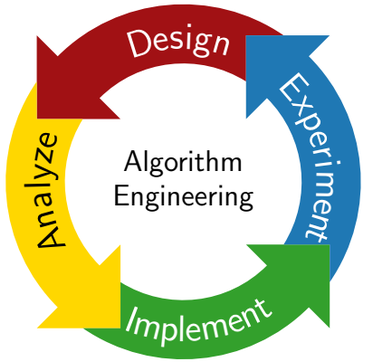
MIP	constraints	variables
full	1,191,406	290,137
callback	21,988	92,681

Example: Sydney &

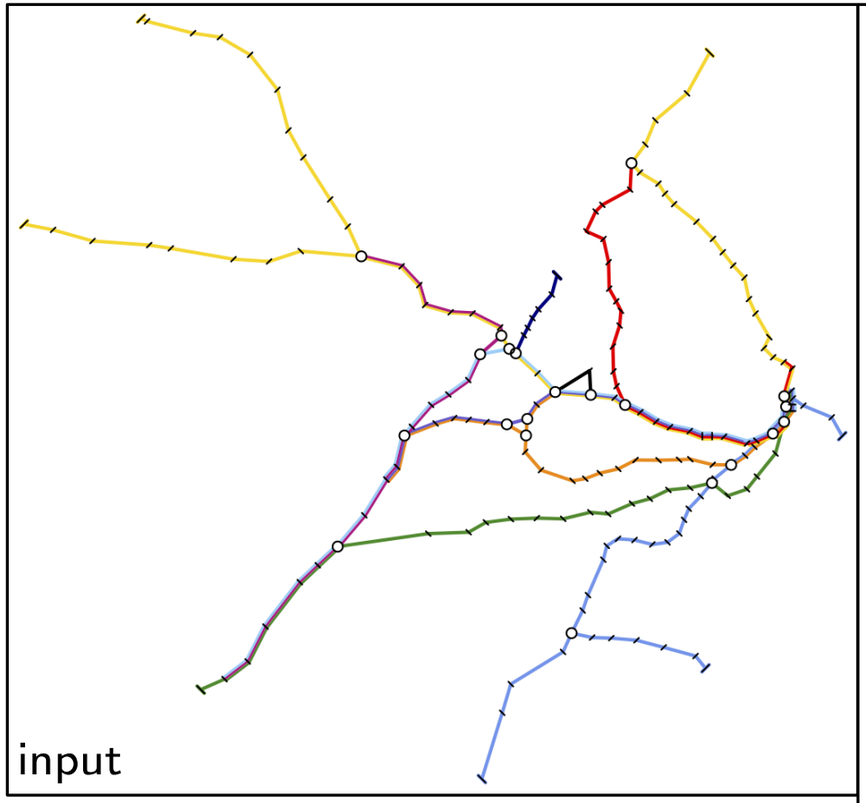


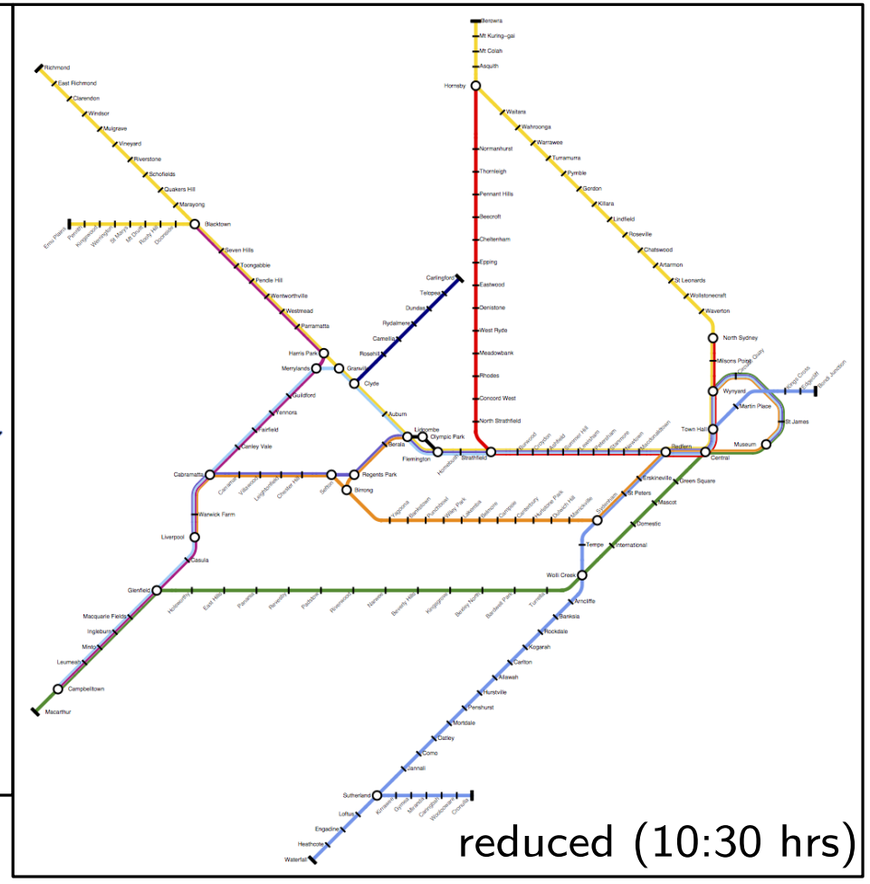
Input	$\lvert V\rvert$	$\lvert E\rvert$	faces	$\lvert\mathcal{L}\rvert$
full	174	183	11	10
reduced	88	97		
labeled	242	270	30	

MIP	constraints	variables
full	1,191,406	290,137
callback	21,988	92,681

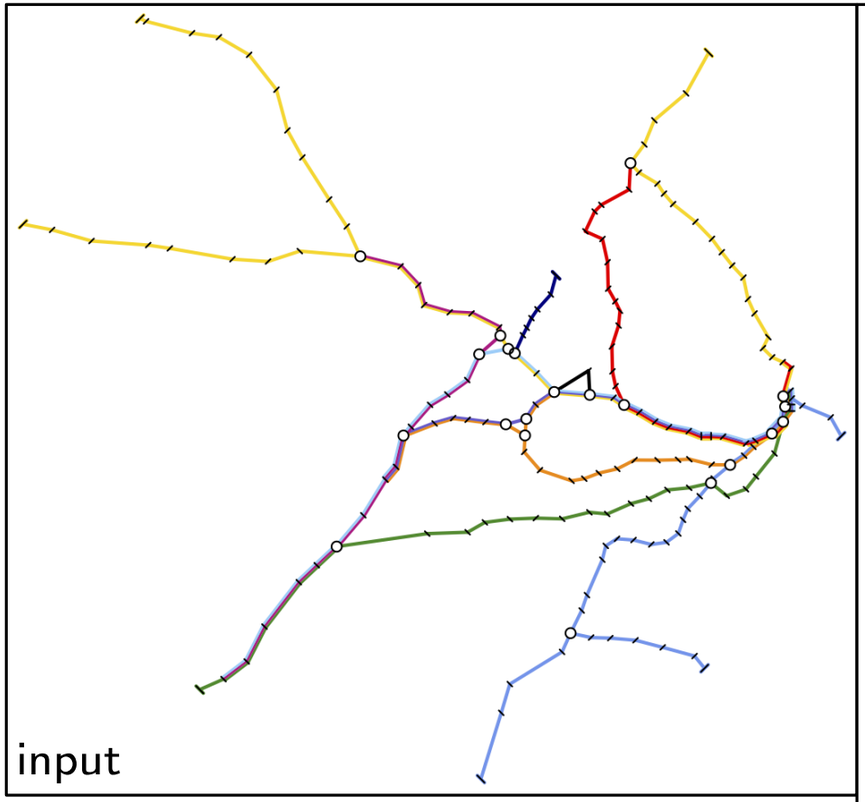


Example: Sydney &

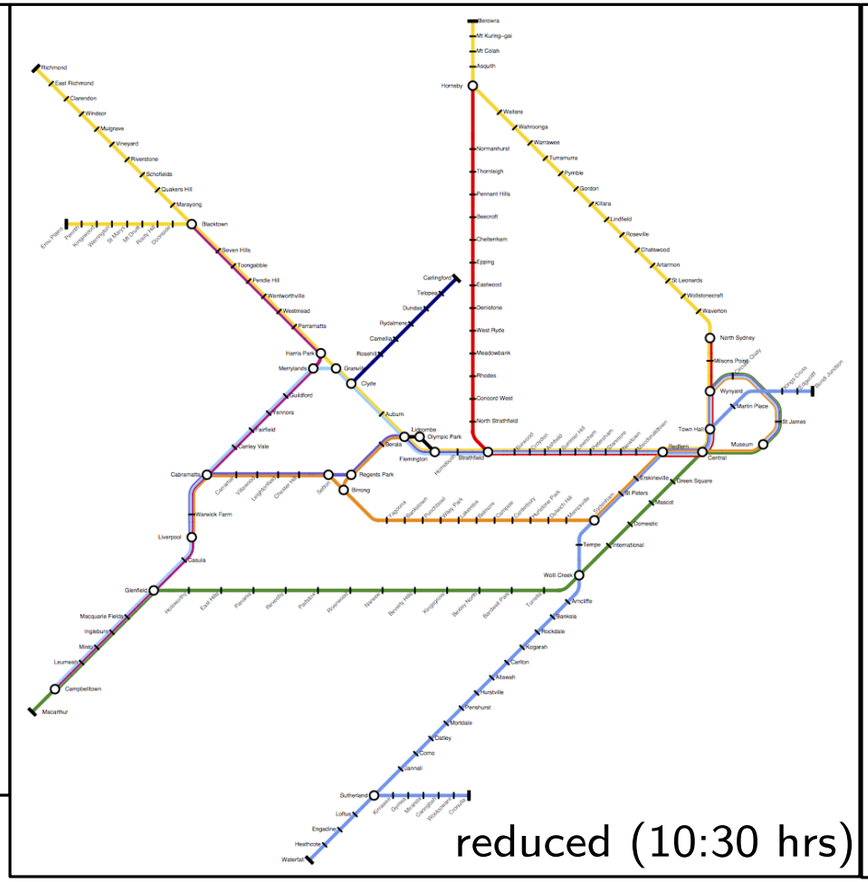

input


reduced (10:30 hrs)

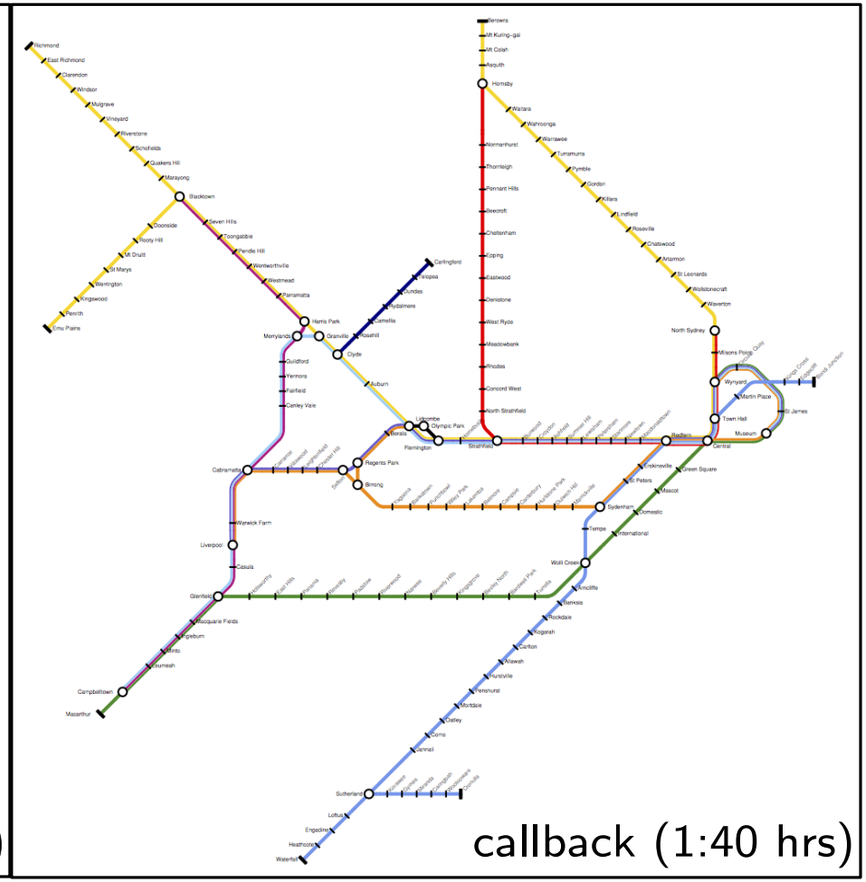
Example: Sydney &



input

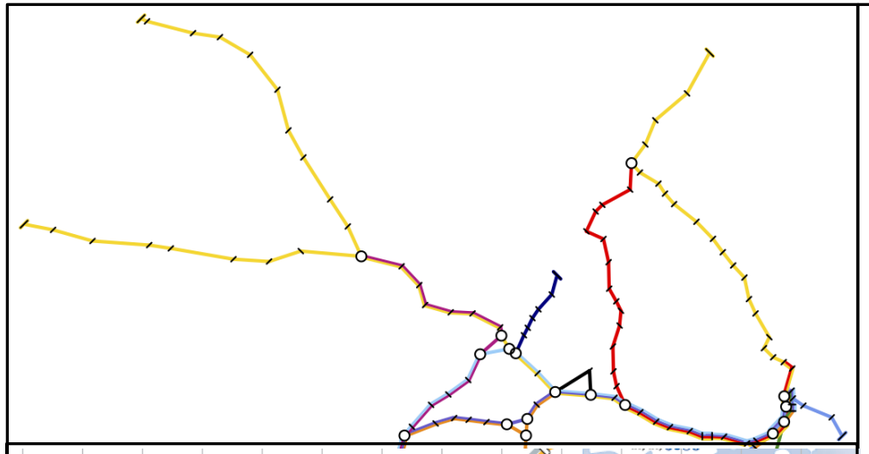


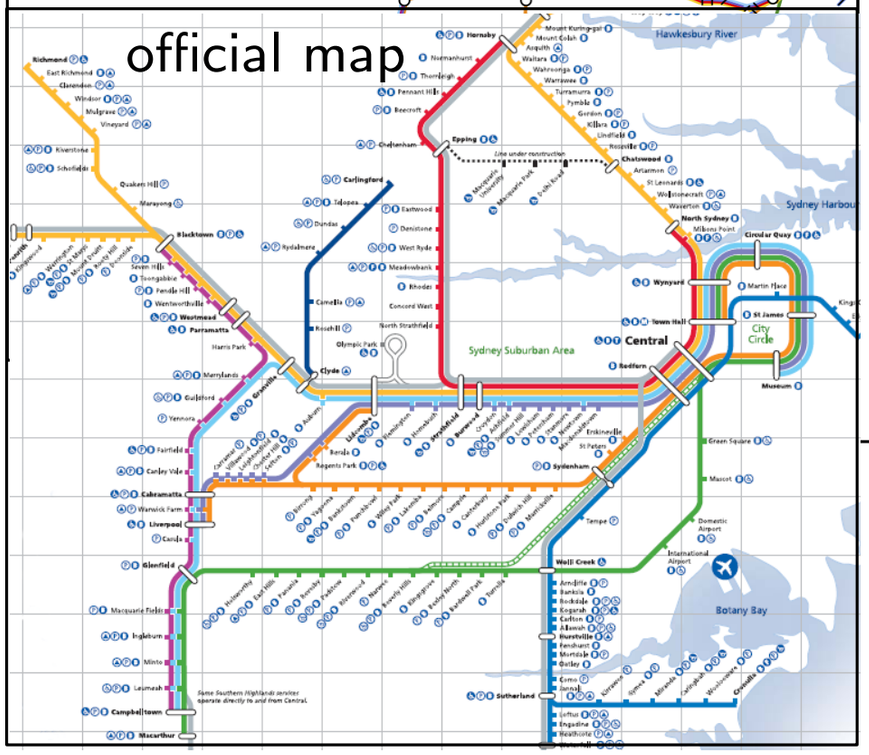
reduced (10:30 hrs)



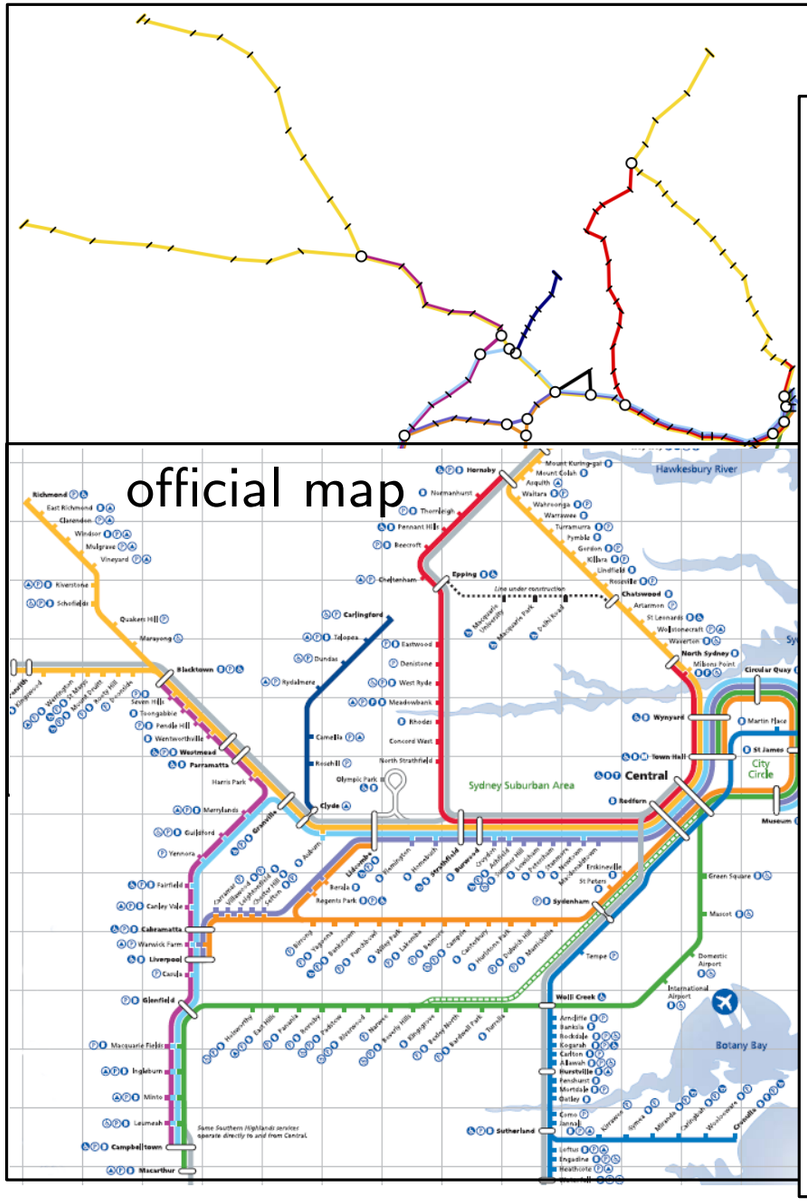
callback (1:40 hrs)

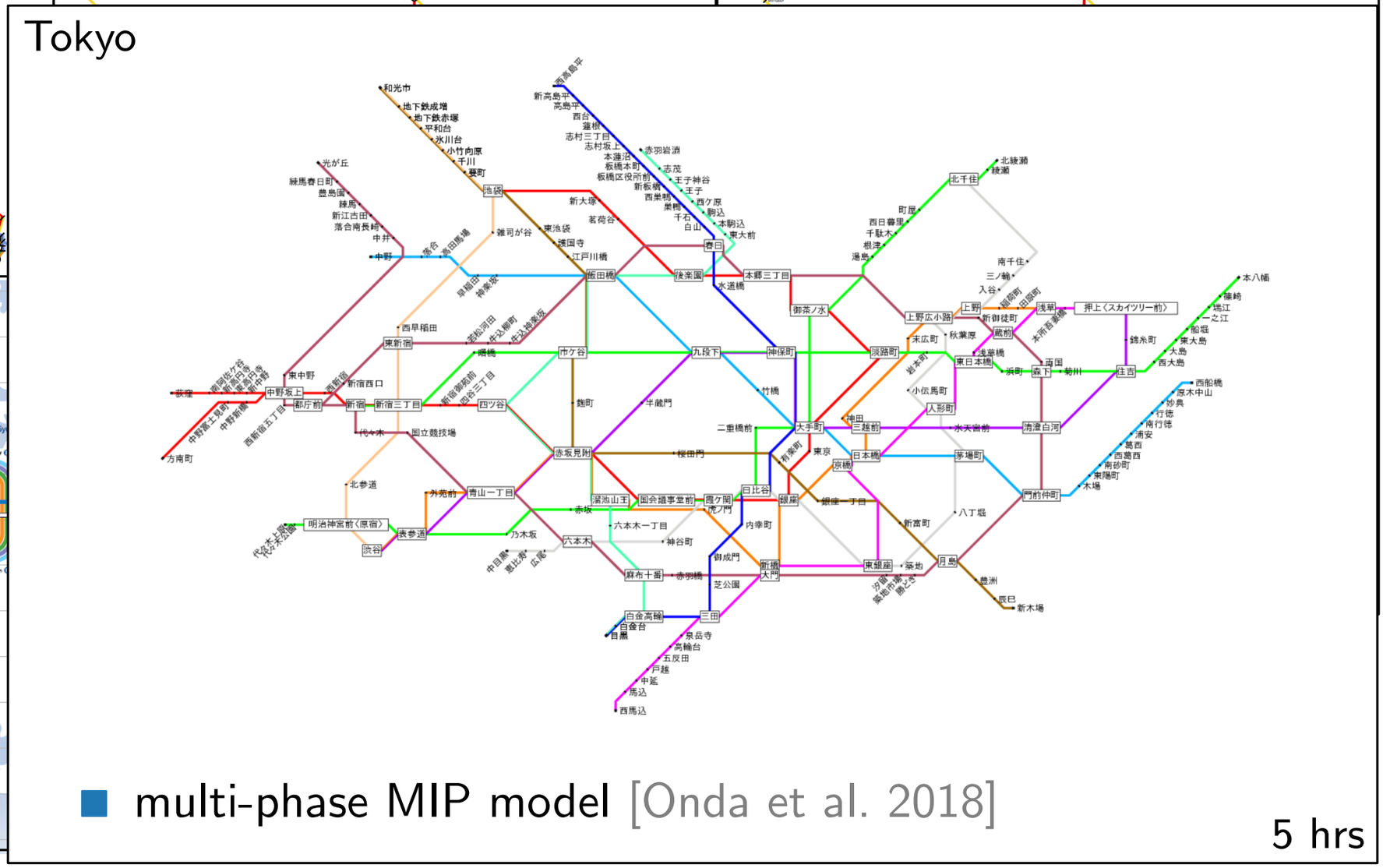
Example: Sydney &




official map
Hawkesbury River
Sydney Harbour
Sydney Suburban Area
Botany Bay
Hornsby
North Sydney
Circular Quay
Wynyard
Town Hall
Central
St James
City Circle
Museum
Redfern
Epping
Chatswood
Carlingford
Parramatta
Blacktown
Liverpool
Cabramatta
Glenfield
Macquarie Fields
Campbelltown
Macarthur
Sutherland
Cronulla
Bondi Junction
Strathfield
Olympic Park
Lidcombe
Regents Park
Green Square
Mascot
Domestic Airport
International Airport
Tempe
Wolli Creek
Hurstville
Kogarah
Rockdale
Arncliffe
Bankstown
Sydenham
Erskineville
St Peters
Newtown
Macdonaldtown
Redfern
Waterloo
Central
Museum
Martin Place
Town Hall
Wynyard
Milsons Point
North Sydney
Waverton
Wollstonecraft
St Leonards
Artarmon
Chatswood
Lindfield
Killara
Gordon
Pymble
Turramurra
Warrawee
Wahroonga
Waitara
Mount Colah
Mount Kuring-gai
Asquith
Hornsby
Thornleigh
Normanhurst
Fox Studios
Darling Harbour
Pyrmont Bridge
Wynyard
Martin Place
Town Hall
Museum
Central
Redfern
Waterloo
Green Square
Mascot
Domestic
Tempe
International
Wolli Creek
Arncliffe
Banksia
Rockdale
Kogarah
Carlton
Allawah
Hurstville
Penshurst
Mortdale
Oatley
Como
Jannali
Sutherland
Loftus
Engadine
Heathcote
Waterfall
Cronulla
Bondi Junction
Strathfield
Olympic Park
Lidcombe
Flemington
Homebush
Concord West
North Strathfield
Rhodes
Meadowbank
West Ryde
Denistone
Eastwood
Epping
Cheltenham
Beecroft
Pennant Hills
Thornleigh
Normanhurst
Warrawee
Wahroonga
Waitara
Mount Colah
Mount Kuring-gai
Asquith
Hornsby
Gordon
Killara
Lindfield
Roseville
Chatswood
Artarmon
St Leonards
Wollstonecraft
Waverton
North Sydney
Milsons Point
Wynyard
Martin Place
Town Hall
Museum
Central
Redfern
Waterloo
Green Square
Mascot
Domestic
Tempe
International
Wolli Creek
Arncliffe
Banksia
Rockdale
Kogarah
Carlton
Allawah
Hurstville
Penshurst
Mortdale
Oatley
Como
Jannali
Sutherland
Loftus
Engadine
Heathcote
Waterfall
Cronulla
Bondi Junction
Strathfield
Olympic Park
Lidcombe
Flemington
Homebush
Concord West
North Strathfield
Rhodes
Meadowbank
West Ryde
Denistone
Eastwood
Epping
Cheltenham
Beecroft
Pennant Hills
Thornleigh
Normanhurst
Warrawee
Wahroonga
Waitara
Mount Colah
Mount Kuring-gai
Asquith
Hornsby
Gordon
Killara
Lindfield
Roseville
Chatswood
Artarmon
St Leonards
Wollstonecraft
Waverton
North Sydney
Milsons Point
Wynyard
Martin Place
Town Hall
Museum
Central
Redfern
Waterloo
Green Square
Mascot
Domestic
Tempe
International
Wolli Creek
Arncliffe
Banksia
Rockdale
Kogarah
Carlton
Allawah
Hurstville
Penshurst
Mortdale
Oatley
Como
Jannali
Sutherland
Loftus
Engadine
Heathcote
Waterfall
Cronulla
Richmond
East Richmond
Clarendon
Windsor
Mulgrave
Vineyard
Riverstone
Schofields
Quakers Hill
Marayong
Blacktown
Seven Hills
Toongabbie
Pendle Hill
Wentworthville
Westmead
Parramatta
Harris Park
Merrylands
Guildford
Yennora
Fairfield
Carlingford
Telopea
Dundas
Rydalmere
Camellia
Rosehill
Clyde
Auburn
Granville
Carlingford
Telopea
Dundas
Rydalmere
Camellia
Rosehill
Clyde
Auburn
Granville
Merrylands
Guildford
Yennora
Fairfield

Example: Sydney & Tokyo





- multi-phase MIP model [Onda et al. 2018]

5 hrs

Mixed-Integer Programming – Discussion

Pros.

Cons.

Mixed-Integer Programming – Discussion

Pros.

- flexible framework, but integration and linearization of new criteria requires some effort
- high layout and labeling quality
- theoretical guarantees
- can integrate user constraints dynamically

Cons.

Mixed-Integer Programming – Discussion

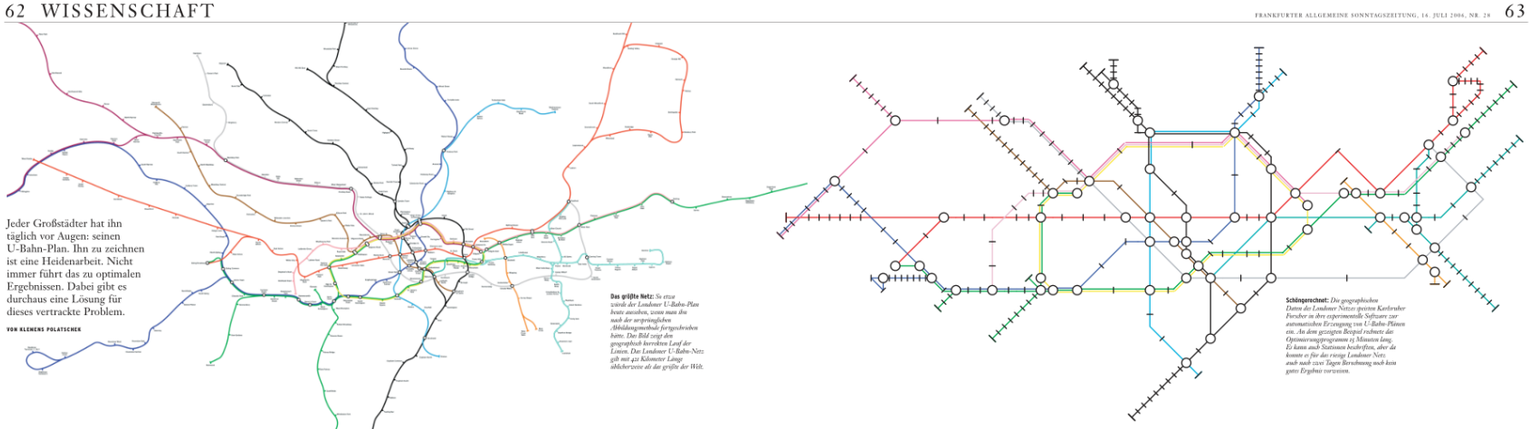
Pros.

- flexible framework, but integration and linearization of new criteria requires some effort
- high layout and labeling quality
- theoretical guarantees
- can integrate user constraints dynamically

Cons.

- long, sometimes unpredictable running times
- for large labeled networks no proof of optimality
- solutions only as good as the model specification

Mixed-Integer Programming – Discussion



Jeder Großstädter hat ihn täglich vor Augen: seinen U-Bahn-Plan. Ihn zu zeichnen ist eine Heidenarbeit. Nicht immer führt das zu optimalen Ergebnissen. Dabei gibt es durchaus eine Lösung für dieses vertrackte Problem.

VON KLEMENS POLATSCHEK

Das größte Netz: *So etwas würde der Londoner U-Bahn-Plan heute aussehen, wenn man die nach der ursprünglichen Abbildungsmethode fortgeschrieben hätte. Das Bild zeigt den geographisch korrekten Lauf der Linien. Das Londoner U-Bahn-Netz gilt mit 402 Kilometer Länge obkilometrie als das größte der Welt.*

Schönrechnet: *Die geographischen Daten des Londoner Netzes speisten Karlsruher Forscher in ihre experimentelle Software zur automatischen Erzeugung von U-Bahn-Plänen ein. An dem gezeigten Beispiel rechnete das Optimierungsprogramm 15 Minuten lang. Es kann auch Stationen beseitigen, aber da konnte er für die riesige Londoner Netz auch nach zwei Tagen Berechnung noch kein gutes Ergebnis vorweisen.*

DIE SCHÖNHEIT DES UNTERGRUNDES

Alexander Wolff ist Informatiker und muß ein glücklicher Mensch sein. Denn seit ein paar Monaten sieht er ein Problem gelöst, das er zu seinem Lebensthema gemacht hat – jedenfalls soweit, wie ein Wissenschaftler in einer Lösung etwas anderes erkennen kann als die Erlösung, neue Stoffen des Ungelösten auszuspähen. Schon Wolffs Diplomarbeit an der Freien Universität Berlin hatte ihn 1992 mit der optimalen Beschriftung von Landkarten aller Art zusammengebracht; promoviert hat er darüber auch. Im Juni 2000 animierte er dann seine Studenten an der Universität Greifswald in einem Seminar: „Ich fühle die Themenhitze mit Problemen, die ich selbst schon immer gern gelöst hätte, aber entweder keinen Erfolg damit hatte oder keine Zeit dafür. Das Thema von Graphendarstellung und Graphenbeschriftung, hier unter dem Begriff U-Bahn-Diagramme formuliert, hat von allen in der Praxis sicherlich die größte Wichtigkeit."

Computererzeugte U-Bahn-Pläne also. Nach seinem Wechsel an die Universität Karlsruhe verschildt Wolff an dieser Frage einen ausgewachsenen Doktoranden und ein paar eigene Blechtöfte, bis er in dem jungen Martin Nöllenburg einen kongenialen Partner fand, der die Frage nicht nur willig schulterte, sondern im Rahmen einer Diplomarbeit auch zu einem fulminanten Ergebnis brachte, wie Wolff lobend erzählt: „Die Arbeit hätte schon so eine Eins gekriegt, aber daß er darin auch noch den mathematischen Beweis schaffte, wie schwierig die Lösung eigentlich ist, das verdiente eine Eins mit Sternchen."

Doch worin besteht diese Schwierigkeit überhaupt? Wir alle sind mit schematischen Netzplänen von U-Bahnen – oder den öffentlichen Nahverkehrs generell – derart vertraut, daß uns das Spezielle daran kaum mehr auffällt. Es ist sogar so: Je besser sie sind, desto weniger wird einem die Qualität bewußt. Denn so ein Diagramm soll eben die mühelose Navigation innerhalb eines Nahverkehrssystems ermöglichen, es also möglichst einfach machen, die passende Station zum eigenen Ziel zu finden, die bunten Linien schnell nachzuverfolgen und alle Umsteigestellen zu sehen, die einen dorthin führen.

Es handelt sich hierbei sicherlich um kein weltmaximierendes Problem. Aber eines zum Verbissen. Und es ist ein Problem für Handgreiflichkeiten, denn für den geschulten Informationsdesigner ist die Detailfreudigkeit ungedehnte Netzschemata ein Debattengegenstand ersten Ranges.

In der Tat werden die großen Pläne bis heute von Hand gezeichnet – natürlich inzwischen am Computer, aber eben gesteuert vom Mausklick des Grafikers. Daß eine Software da systematisch mitzeichnen könnte, ist erst seit kurzem ein Thema. Meist nehmen sich theoretische Informatiker wie Wolff und Nöllenburg der Fragestellung an. Theoretische Informatiker sind häufig nicht so theoretisch, wie es klingt. Das Ruhy liegt nur meist vor ihrer Tür, weil Pläne aller Art sich eben als Bilder von etwas betrachten lassen, was der Mathematiker „Graphen" nennt. Das sind Punkte, die durch Linien, oder genauer gesagt „Kanten", verbunden sind. Für solche Graphen gibt es eine Graphentheorie, also eine Menge an mathematischen Formeln und Algorithmen, ohne deren Kenntnis man bei konkreten

Schon ein scheinbar simples U-Bahn-Schema ist ein fetter Brocken für den Computer.

Problemen dieser Sorte kaum weiterkommt. Tatsächlich wird aus der Erzeugung eines scheinbar simplen U-Bahn-Schemas ein fetter Klumpen an Optimierungsaufgabe, an der auch ein schneller Computer schier endlos kauen kann.

Der Ausgangspunkt eines jeden U-Bahn-Plans sind ja immer die echten Schienen, auf denen Züge fahren, also die geographische Realität. Wie uns Geographie ein Schema wird, und zwar ein möglichst übersichtliches, das ist die Frage, um die es hier geht. Und jeder, der sich ernsthaft mit ihr befaßt, lernt als erstes die große heimeische Wende kennen, nämlich die freuden- und tränenreiche Geschichte des technischen Zeichners Henry C. Beck, dessen revolutionärer Londoner U-Bahn-Plan von 1933 ein System etablierte, das sich die Welt so hefig zu eigen machte, daß wir Heutigen darin gar nichts Besonderes mehr erkennen (siehe „Der Mann, der die Nudeln geradezog").

Neben Alexander Wolff und Martin Nöllenburg haben sich seit ein paar Jahren noch einige Graphenforscher den U-Bahn-Plänen zugewandt. Wie in jeder Subdisziplin der Wissenschaft kennt man sich über die Kontinente hinweg, von Konferenzen und aus der Fachliteratur. Und alle kennen den Beckschen Urgrund. Wenn man aber in einem internationalen Vergleichstest die Ergebnisse nebeneinander hält, dann scheint es, als hätten nur Wolff und Nöllenburg am Ende die Nerven gehabt, bis ganz nach unten zu stochern, wo die ästhetische und praktische Essenz der Sache liegt – der U-Bahn-Plan als Prinzip.

Vielleicht beantwortet gerade ein so scheinbar entlegenes Feld am besten die Frage, über die auch im deutschen Informatikjahr 2006 das öffentliche Kopfzerbrechen anhält, nämlich wie Informatiker eigentlich ticken und wozu genau sie nütze sind. Denn die Karlsruher Arbeit zeigt, auf wie wenigen Regeln unser ästhetischer und wahrnehmungspsychologisch-ergonomischer Klodderdatsch manchmal beruht. Und sie beweist, daß er technisch nachzubauen ist. Schönheit wird zur Rechengröße! Interessant ist natürlich auch, wo dieser Weg seine Grenzen findet. Ein funktionierendes System nach Karlsruher Art wird die Grafikdesigner nicht arbeitslos machen. Das war nicht das Ziel, und die entstandene Software taugt heute allenfalls für gute Vorlagen, nicht für die Feinausführung.

Doch denke man an die vielen schicken Pläne, die in Zukunft ad hoc nach Nutzervorgaben, im Detail, als Ausschnitt, live und online, vom Handy und fürs Web zu berechnen wären.

Als Wolff und Nöllenburg ihren himorischen Henry Beck zu Ende studiert und lange über einem Stapel moderner U-Bahn-Pläne gegrübelt hatten, erdachten sie sieben schlichte Regeln, die aus einem geographischen Plan ein Netzschema machen. „Das Problem so zu modellieren war eigentlich die Hauptarbeit", berichten sie. Von den sieben waren es zunächst vier harte Regeln, denen die neue „schöne Zeichnung", wie sie das nennen, ohne Ausnahme folgen muß: Sie muß der geographischen Vorlage topologisch gleichen, die heißt, jeder Knoten muß rundum wieder mit den gleichen Knoten durch je eine Kante verbunden sein. Jede Kante muß außerdem mit einer gewissen festgelegten Mindestlänge abgebildet werden.

Jeder Knoten muß darüber hinaus einen festgelegten Mindestabstand von vorbestimmten Kanten halten. Und schließlich muß die neue Zeichnung „oktilinear" werden. Jede Kante muß von jedem Knoten weg ausnahmslos in eine von acht Richtungen weisen – horizontal, vertikal oder diagonal entweder 45 Grad nach links oder nach rechts gegrad. Aus war seinerzeit der große Kniff des Henry Beck.

Die drei weiteren Regeln sind eher weich. Das heißt: Je besser eine neue Zeichnung sie erfüllt, ein-mem-desto-höheren Schönheitswert bekommt sie in der darauf berechneten Optimierung zugeteilt. Erstens soll jede U-Bahn-Linie darin möglichst wenige Knicke aufweisen, also jede ihrer Kanten möglichst in dieselbe Richtung weisen wie ihre Nachbarn auch. Zweitens sollte die Summe sämtlicher Kantenlängen in der Zeichnung möglichst klein sein (je kompakter das Ganze, desto hübscher). Drittens sollte jede Kante möglichst nahe der Richtung liegen, die sie in der ursprünglichen geographischen Abbildung hat.

Die Bilder, die der Computer nach diesen sieben Regeln am Ende auswirft, ähneln en gros wie en detail oft verblüffend den menschengemachten. Man versteht auf einmal, wovon die zahllosen Skizzenzettel handelten, mit denen Henry Beck über Jahrzehnte hinweg sein Heim pflasterte. Das waren ebenfalls Optimierungsläufe, gekreuzte verbesserte Lösungen. Denn wenn Regeln einmal auf die beschriebene Weise formuliert sind, kann man sie für ohne Umstände einem der verfügbaren Standardprogramme für eine sogenannte gemischt-ganzzahlige Optimierung vorwerfen. Dieses Programm wirft dann einen optimal schönen Plan aus.

Man muß nur manchmal Geduld haben. Mathematisch gehört die U-Bahn-Aufgabe nämlich zu einer ebenso häufigen wie unbeliebten Klasse von Rechenproblemen, die man „NP-vollständig" nennt. Das ist der Beweis, den Martin Nöllenburg führte. Nach heutiger Erkenntnis gibt es bei Problemen dieser Klasse keinen Trick für schnelle Lösungen. Es hilft nur schiere Rechenkraft, weshalb umfangreiche Aufgaben sich in der Praxis einer präzisen oder garantierten Lösung verweigern.

Damit taucht plötzlich ein mathematischer Grund dafür auf, warum Henry Beck in London viele Jahre mit dem Versuch zubrachte, möglichst wenige U-Bahn-Linien als Diagonalen abzubilden. Je mehr Kanten weg ausnahmslos in eine von acht Richtungen weisen – horizontal, vertikal oder diagonal entweder 45 Grad nach links oder nach rechts gegrad. Aus war seinerzeit der große Kniff des Henry Beck.

Die drei weiteren Regeln sind eher weich. Das heißt: Je besser eine neue Zeichnung sie erfüllt, ein-mem-desto-höheren Schönheitswert bekommt sie in der darauf berechneten Optimierung zugeteilt. Erstens soll jede U-Bahn-Linie darin möglichst wenige Knicke aufweisen, also jede ihrer Kanten möglichst in dieselbe Richtung weisen wie ihre Nachbarn auch. Zweitens sollte die Summe sämtlicher Kantenlängen in der Zeichnung möglichst klein sein (je kompakter das Ganze, desto hübscher). Drittens sollte jede Kante möglichst nahe der Richtung liegen, die sie in der ursprünglichen geographischen Abbildung hat.

Die Bilder, die der Computer nach diesen sieben Regeln am Ende auswirft, ähneln en gros wie en detail oft verblüffend den menschengemachten. Man versteht auf einmal, wovon die zahllosen Skizzenzettel handelten, mit denen Henry Beck über Jahrzehnte hinweg sein Heim pflasterte. Das waren ebenfalls Optimierungsläufe, gekreuzte verbesserte Lösungen. Denn wenn Regeln einmal auf die beschriebene Weise formuliert sind, kann man sie für ohne Umstände einem der verfügbaren Standardprogramme für eine sogenannte gemischt-ganzzahlige Optimierung vorwerfen. Dieses Programm wirft dann einen optimal schönen Plan aus.

Man muß nur manchmal Geduld haben. Mathematisch gehört die U-Bahn-Aufgabe nämlich zu einer ebenso häufigen wie unbeliebten Klasse von Rechenproblemen, die man „NP-vollständig" nennt. Das ist der Beweis, den Martin Nöllenburg führte. Nach heutiger Erkenntnis gibt es bei Problemen dieser Klasse keinen Trick für schnelle Lösungen. Es hilft nur schiere Rechenkraft, weshalb umfangreiche Aufgaben sich in der Praxis einer präzisen oder garantierten Lösung verweigern.

Die Oktilinearität gleichgültig oder weniger wichtig – ästhetischer Wert entsteht so höchstens zufällig. (Siehe „Der Grand Prix von Sydney".)

Die Karlsruher Software läßt um gleichlerweise auch nicht teilgelung auf eine Optimierung warten. Bereits während des Laufs lassen sich neue Kriterien einrechnen. Das nutzen Wolff und Nöllenburg für den Trick, die Modell nicht mit einer generellen Überprüfung der Frage zu belasten, ob sich irgendwelche Linien in der Zeichnung überschneiden. Interessierte Kandidaten für eine schöne Zeichnung werden auf scheinbare Linien gepriift, und eine solche unerlaubte Überlappung wird durch Setzen eines einzelnen zusätzlichen Kriteriums ausgeschlossen. Der Nutzer kann die Kandidaten jederzeit betrachten und wird in den meisten Fällen mit den ersten guten Lösungen zufrieden sein, auch wenn die noch gar nicht rechnerisch optimal sind. Für das vergleichsweise wenig verknotete Wiener U-Bahn-Netz etwa lieferte die Karlsruher Software auf einem haushaltsüblichen PC schon nach ei Sekunden eine erstklassige Lösung.

Die Software kann die U-Bahn-Stationen auf den Plänen auch beschriften. Das ist eigentlich ein eigener Zweig der Forschung und keine geringe Kunst. Mathematisch gesehen erhoht die Vorgabe, möglichst große (aber einheitlich große) Texte ohne Überschneidungen in ein Diagramm zu plazieren, die Komplexität der Aufgabe dramatisch. Doch lösen die Karlsruher die Anforderung durch einen eleganten Kniff. Beschriftungen werden durch eine Art Pseudo-U-Bahn umhüllt, die dann in der Gesamtoptimierung mitläuft. Weil dabei so viel mehr Eingabekriterien vorliegen, dauert die Suche nach einer schönen Zeichnung länger. Im Fall Wien berichtet Nöllenburg von 432 Stunden und ergänzt: „Die ersten vergleichbar guten Lösungen tauchten aber schon nach 14 Stunden auf, und machbare Lösungen schlechter Qualität gab es auch nur dreieinhalb Minuten."

Für die verzweigten Netze von Sydney und London fand sich mit zwanzigen Rechenläufen noch gar kein schön beschriftbarer Plan. Hier bedarf die Optimierung also noch der Optimierung – oder einer massiv verstärkten Rechenkapazität. Auch im Detail mißfin die Verschönerungen jetzt noch manchmal. Umsteigepunkte anders zu markieren, den Linien eine beliebige Breite geben zu können. Wenn darf man damit rechnen! „Wir werden das Thema wohl nicht weiterverfolgen", sagt Martin Nöllenburg. Alexander Wolff, der Anfang des Jahres in Karlsruhe habilitierte, nicht gerade weiter auf eine neue Stelle nach Eindhoven. „Das Programmierende werden wir wohl freigeben, damit er nicht verschimmelt", sagt er. Vielleicht gibt es ja irgendwo jemanden, der nicht weiter U-Bahn-Pläne von Hand zeichnen will.

Quellen: „M-Berlin's Underground Map" von Ken Garland (1994); „Underground Maps After Beck" von Mark Ovenden (2003); „Metro Maps of the World" von Mark Ovenden (2005); alle bei Capital Transport Publishing in London erschienen. **Website:** www.urtan-rak.net und www.mi-ro.com/metro



Das Original: *Henry Becks erster Entwurf für die Londoner U-Bahn, veröffentlicht im Januar 1933 – heute ein Meilenstein der Designgeschichte. Schematische Darstellungen einzelner Strecken gab es zwar damals zum Beispiel schon in der U-Bahn-Hagen, aber auf ein Gesamtschema war vorher nie jemand gekommen.*



Deutsche Revolution: *Außerhalb Englands findet man ein Diagramm, das Becks Ideen komplett vorwegnimmt. Es ist ein Berliner Netzschema von 1931, perfekt oktilinear aufgebaut, vom S-Bahn-Ring abgesehen. Der entscheidende Unterschied zu London: Dort wandte eine bislang Tradition und ein weitreichendes Verbild geschaffen. In Berlin vergaß man die Vorzüge des Schemas bald wieder und kehrte lange vor geographischen Karte zurück. Man hatte wohl mit anderen Umständen zu tun.*



Verwirrung in Manhattan: *Die Neu Yorker U-Bahn gilt nach London als zweitgrößte, es hat sogar mehr Stationen, 1972 fuhrte es diesen neuen „Subway Guide" nach dem Muster von Beck ein, gestaltet vom Star-Designer Massimo Vignelli. Der Plan wurde wieder abgeschafft. Die New Yorker fanden die Abweichungen von der Geographie zu irritierend.*

Der Mann, der die Nudeln geradezog

Mit Henry Beck begann die Ära der aufgeräumten Diagramme

Wer das U-Bahn-Wesen generell, so begannen sich nach seine Dienstjungsprobleme: in London. Man schrieb 1908, da waren die damals privaten Londoner U-Bahnen in Not. Ihre sinnlerare Maulwurfnetz in der Londoner Innenstadt war abgeschlossen, man mußte neue Passagiere anlocken. Deshalb gaben die Konkurrenten erstmals einen gemeinsamen Plan heraus. Diese geographische Karte zeigte den Verlauf der Linien unter Stadtteilen und Straßen.

Bald wuchsen die Linien aber weiter hinaus in die Vororte. Auf dem fortgeschrittenen Plan ewiggte sich das wichtige Innenstadt-Netz auf so knapper Fläche zusammen, daß Nützliches kaum noch zu erkennen war. Die Zeichner hielen alle Einzelheiten weg, sie fingen sogar widerwillig an, der Karte einen gleitenden Mittelab zu geben, im Zentrum größer als außen. Trotzdem erinnerte der Plan schließlich nur noch an verwickelte Spaghetti beschreibungsweise ihre noch dünnere Variante „Vermicelli", wie Henry Beck sie nannte.

Henry C. Beck („Harry" unter Freunden) wurde der Mann, der die Nudeln geradezog. Geboren Anfang des Jahrhunderts, fing er 1915 als junger technischer Zeichner bei der U-Bahn zu arbeiten an. Die Jahre waren hart, er bekam nur einen Zeitarbeitsvertrag; mal beschäftigt, mal gekündigt, die Jungsten traf es immer zuerst.

In seiner freien Zeit, irgendwann 1931, traf den Henry Beck vor dem Vermicelli-Plan ein Geistesblitzkrieg. Warum nicht die Geographie beiseite lassen! Die scherte den Untergrundexpedition je gar nicht. Alle Linien geradezichen und aufräumen, sie horizontal und vertikal verteilen wie auf einem elektrischen Schaltplan, vielleicht auch Diagonalen, die Stationen gleichmäßig auf die Streckn vertelen – das war es doch! Ein übersichtliches Diagramm statt einer Karte.

Im Sommer 1931 zeigte er der Werbeabteilung seines Unternehmens den Entwurf. Kein Interesse, hieß es. Zu revolutionär. Henry Beck steckte das Blatt in die Schublade, zog es ein Jahr später wieder hervor und wagte einen neuen Anlauf. Überraschung: Nun wollte man es versuchen!

Der kleine Fehlplan wurde im Januar 1933 gedruckt und vom Publikum ins Herz geschlossen. Die erste 700 000 Exemplare waren rasch vergriffen, im Februar mußten schon 900 000 nachgedruckt werden. Im Sommer 1933 stellte die gesamte gerade integrierte U-Bahn ihre Plakate auf das Schema Beck um.

1934 verfaßte der einflußreiche britische Designer Ken Garland das Standardwerk über Becks Arbeiten. Er wollte der Genialmeisterung seines Freundes ein Denkmal setzen und stellte fest: „Es wird sonst nur wenige Einzelarbeiten geben, die so bahnbrechend und dauerhaft maßgebend waren." Es stimmt schon. Wie leicht zu erkennen, haben Nah-







verkehrsnetze in aller Welt sich die Idee angeeignet.

Und Beck? Für Entwurf und Arbeit, die Monate der Mühen, bekam er 20,20 Pfund als Honorar, was man sich in heutiger Kaufkraft als 1000 Euro vorstellen darf. Doch Henry Beck saß doppelt in der Falle. Wegen seines prekären Arbeitsverhältnisses gestand man ihm die Idee zu, hielt ihn aber weiterhin wie einen Domestiken. Schlimmer noch: Die Idee selbst nahm Beck in Heft. Jahrelang verbrachte er mit der Fortschreibung des Netzplans in mäßiger Detailverwesenheit. Mindestens einmal zeichnete er aus eigenem Antrieb die ganze Druckvorlage neu, nur um die Biegungen der Linien etwas enger zu fassen. Dazu kamen die Ideen der wechselnden Vorsteher der U-Bahn. Mal mußte Beck die Umsteigsknoten in Form von Kreisen zeigen statt als Raute, dann mußte er Stationsnamen in vorgeschriebene Rasten schreiben oder Punkte in die Kreise setzen, wo auch ein Fettbalsbrief lag. Mal wurden die zentralen Linien, dann alle Linien fetter und ihre Namen in der Linie verwischt, und schließlich alles wieder rerour, weil Beck den Ideen seiner Vorgesetzte Ausführung ihren Lauf ließ.

Ken Garland vermutet, daß sich Beck dem beugte, um bloß seine Arbeit by fest in Händen behalten zu können. Die Neugemeinschaft half, obwohl manche in der Werbeabteilung Beck nur als schwierigen und verdrießlichen Menschen sahen. Im April 1960 dispierte ihn deren Chef Harold F. Hutchinson und veröffentlichte mit Fanfarenstoß einen komplett neu gestalteten Plan, den er selber angefertigt hatte. Er war bei näherer Betrachtung eine amateurhafte Karikatur der Beckschen Ideen. Nach zwei quälenden Jahren wurde das neue Schema nach Beckschen Prinzipien gebeßt und bis heute fortgeschrieben.

Beck selber kam trotz verweifelter Bittbriefe nie mehr zum Zug. Er starb 1974. Heute gilt er als großer Meister, gekürt mit einer eigenen Ausstellung im London Transport Museum, Anklicken von Entwürfen aus seiner Hand und zuletzt, in einer Publikums-Show der BBC, der Wahl seines revolutionären U-Bahn-Plans zur zweitwichtigsten Design-Ikone überhaupt – nach der Concorde. *ky*



Least-Squares Schematization

[Wang, Chi '11]

Idea.

- model layout problem as minimization of a set of (squared) energy terms
- variables for vertex positions and edge slopes
- use iterative numerical optimization method

Least-Squares Schematization

[Wang, Chi '11]

Idea.

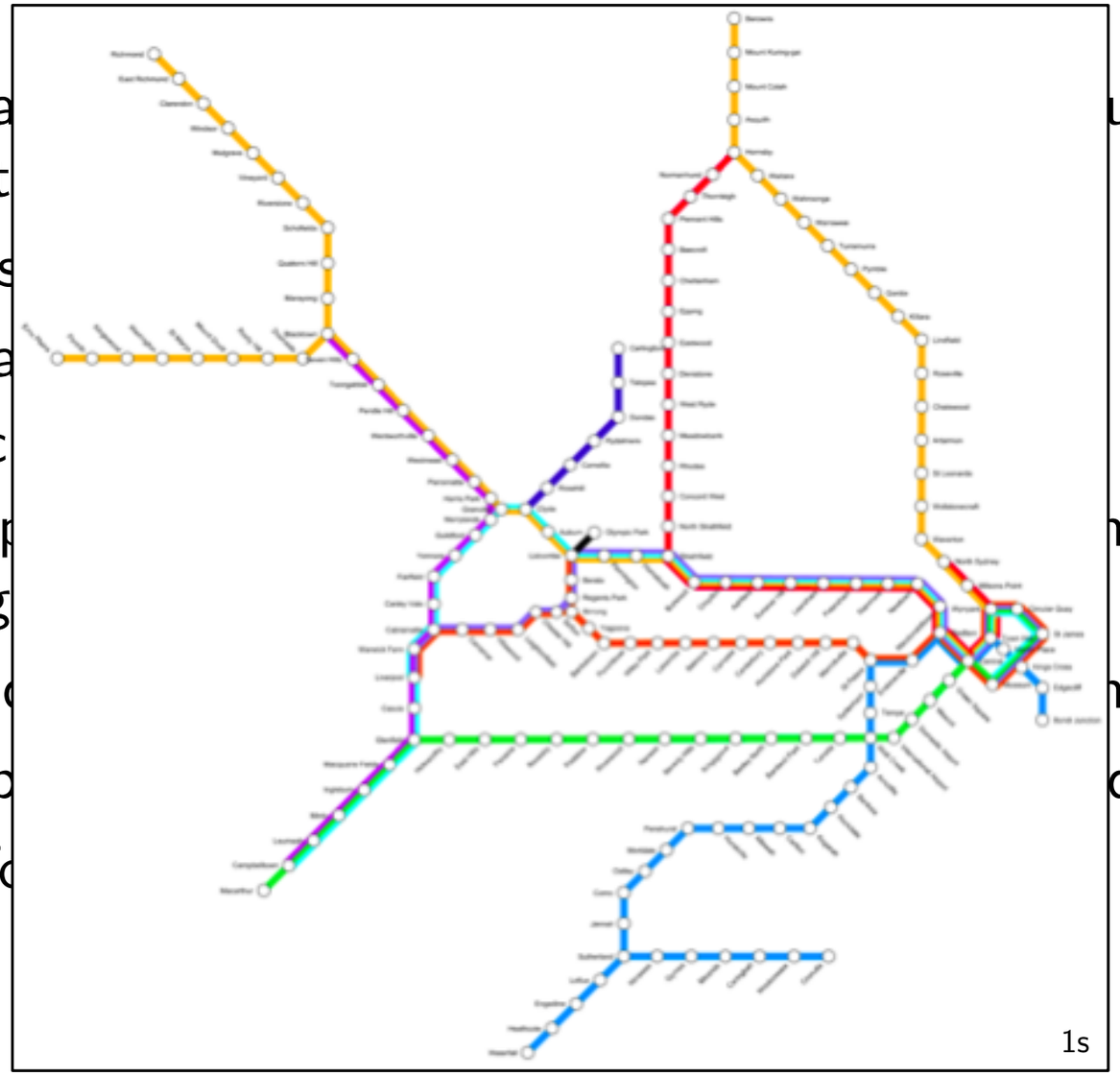
- model layout problem as minimization of a set of (squared) energy terms
  - variables for vertex positions and edge slopes
  - use iterative numerical optimization method
- 3-step approach
  - compute topologically correct non-octilinear layout optimizing angular resolution **(R5)**, uniform edge lengths **(R7)**, displacement **(R6)**
  - octilinearly discretize edge orientations **(R2)** by extra energy term
  - optimize label placement by energy minimization in fixed layout
- described for focus route, generalizes to entire maps

Least-Squares Schematization

[Wang, Chi '11]

Idea.

- model la … uared)
  - energy t …
  - variables …
  - use itera …
- 3-step approac …
  - compute top … mizing angular resolution **(R5)**, uniform edg …
  - octilinearly … nergy term
  - optimize lab … d layout
- described for fo …



Least-Squares Schematization

[Wang, Chi '11]

Idea.

- model layout problem as minimization of a set of (squared) energy terms
  - variables for vertex positions and edge slopes
  - use iterative numerical optimization method
- 3-step approach
  - compute topologically correct non-octilinear layout optimizing angular resolution **(R5)**, uniform edge lengths **(R7)**, displacement **(R6)**
  - octilinearly discretize edge orientations **(R2)** by extra energy term
  - optimize label placement by energy minimization in fixed layout
- described for focus route, generalizes to entire maps

Discussion.

- very fast method for good quality layouts
- no guarantee on constraints unless final energy is zero

Summary

- variety of layout methods evolved over the last 15–20 years
- many shared design rules
- trade-off between speed and quality, but quite reasonable maps can be computed in a matter of seconds to minutes
- many approaches are customizable and open to new criteria
- current trend: beyond octilinear metro maps

Summary

- variety of layout methods evolved over the last 15–20 years
- many shared design rules
- trade-off between speed and quality, but quite reasonable maps can be computed in a matter of seconds to minutes
- many approaches are customizable and open to new criteria
- current trend: beyond octilinear metro maps

Why automated maps?

Summary

- variety of layout methods evolved over the last 15–20 years
- many shared design rules
- trade-off between speed and quality, but quite reasonable maps can be computed in a matter of seconds to minutes
- many approaches are customizable and open to new criteria
- current trend: beyond octilinear metro maps

Why automated maps?

- base layouts for graphic designers (semi-automated process)
- large quantities of individual or special-purpose maps

Summary

- variety of layout methods evolved over the last 15–20 years
- many shared design rules
- trade-off between speed and quality, but quite reasonable maps can be computed in a matter of seconds to minutes
- many approaches are customizable and open to new criteria
- current trend: beyond octilinear metro maps

Why automated maps?

- base layouts for graphic designers (semi-automated process)
- large quantities of individual or special-purpose maps

Challenges

Summary

- variety of layout methods evolved over the last 15–20 years
- many shared design rules
- trade-off between speed and quality, but quite reasonable maps can be computed in a matter of seconds to minutes
- many approaches are customizable and open to new criteria
- current trend: beyond octilinear metro maps

Why automated maps?

- base layouts for graphic designers (semi-automated process)
- large quantities of individual or special-purpose maps

Challenges

- global quality criteria like harmony, coherence, balance
- edge bundles and large vertices

Literature

- [Nöllenburg '14] A Survey on Automated Metro Map Layout Methods
- [Dwyer, Hurst, Merrick '08] A fast and simple heuristic for metro map path simplification
- [Delling et al. '14] On d-regular schematization of embedded paths
- [Brandes & Pampel '09] On the Hardness of Orthogonal-Order Preserv-ing Graph Drawing
- [Gema et al. '11] On d-Regular Schematization of Embedded Paths
- [Hong et al. '06] Automaticvisualisation of metro maps
- [Chivers & Rodgers '14] Octilinear Force-Directed Layout with Mental MapPreservation for Schematic Diagrams
- [Fink, Haverkort, Nöllenburg, Roberts, Schuhmann, Wolff '12] Drawing metro maps using Bézier curves
- [Avelar & Müller '00] Generating topologically correct schematicmaps
- [Ware et al. '06] Automatedproduction of schematic maps for mobile application
- [Ware & Richards '13] An ant colony system algorithm forautomatically schematizing transport network data set
- [Stott et al. '11] Auto-matic metro map layout using multicriteria optimization
- [Nöllenburg & Wolff '11] Drawing and labeling high-quality metro maps by mixed-integer programming
- [Onda, Moriguchi, Imai '18] Automatic Drawing for Tokyo Metro Map
- [Wang & Chi '11] Focus+context metro maps

Oral Exams

- Focus on **methods** such as

Oral Exams

- Focus on **methods** such as
  - (integer) linear programming
  - flow formulations
  - canonical order
  - Schnyder realizer
  - 2-SAT
  - (planar) 3-SAT for NP-hardness proofs

Oral Exams

- Focus on **methods** such as
  - (integer) linear programming
  - flow formulations
  - canonical order
  - Schnyder realizer
  - 2-SAT
  - (planar) 3-SAT for NP-hardness proofs
- Be prepared to "run" an algorithm on an example instance.

Oral Exams

- Focus on **methods** such as
  - (integer) linear programming
  - flow formulations
  - canonical order
  - Schnyder realizer
  - 2-SAT
  - (planar) 3-SAT for NP-hardness proofs
- Be prepared to "run" an algorithm on an example instance.
- You may pick your favorite topic for the first few questions.