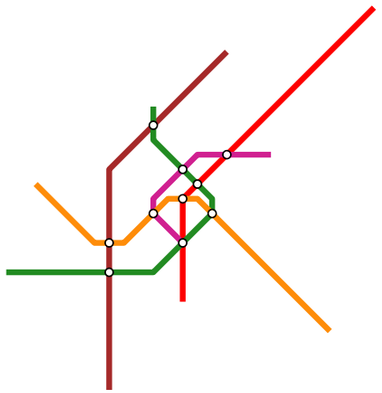
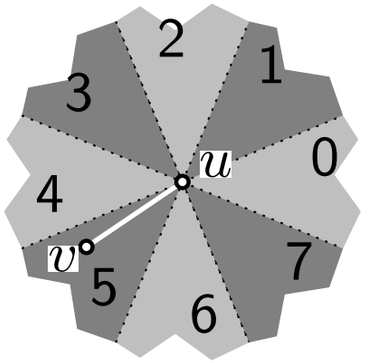
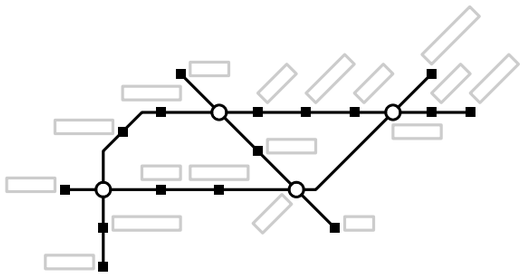
Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Visualization of Graphs



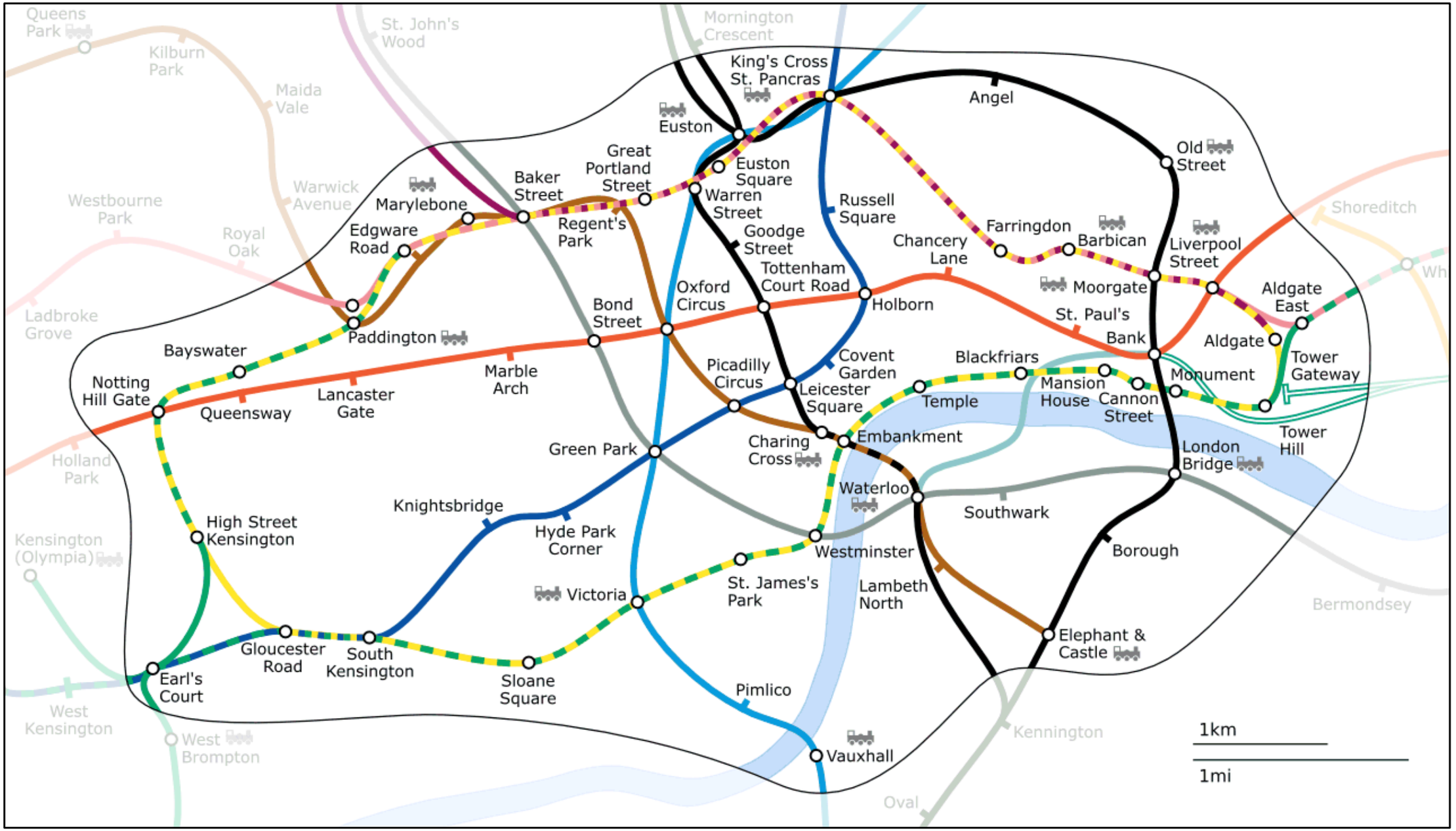
Lecture 12:
Octilinear Graph Drawing
Metro Map Layout



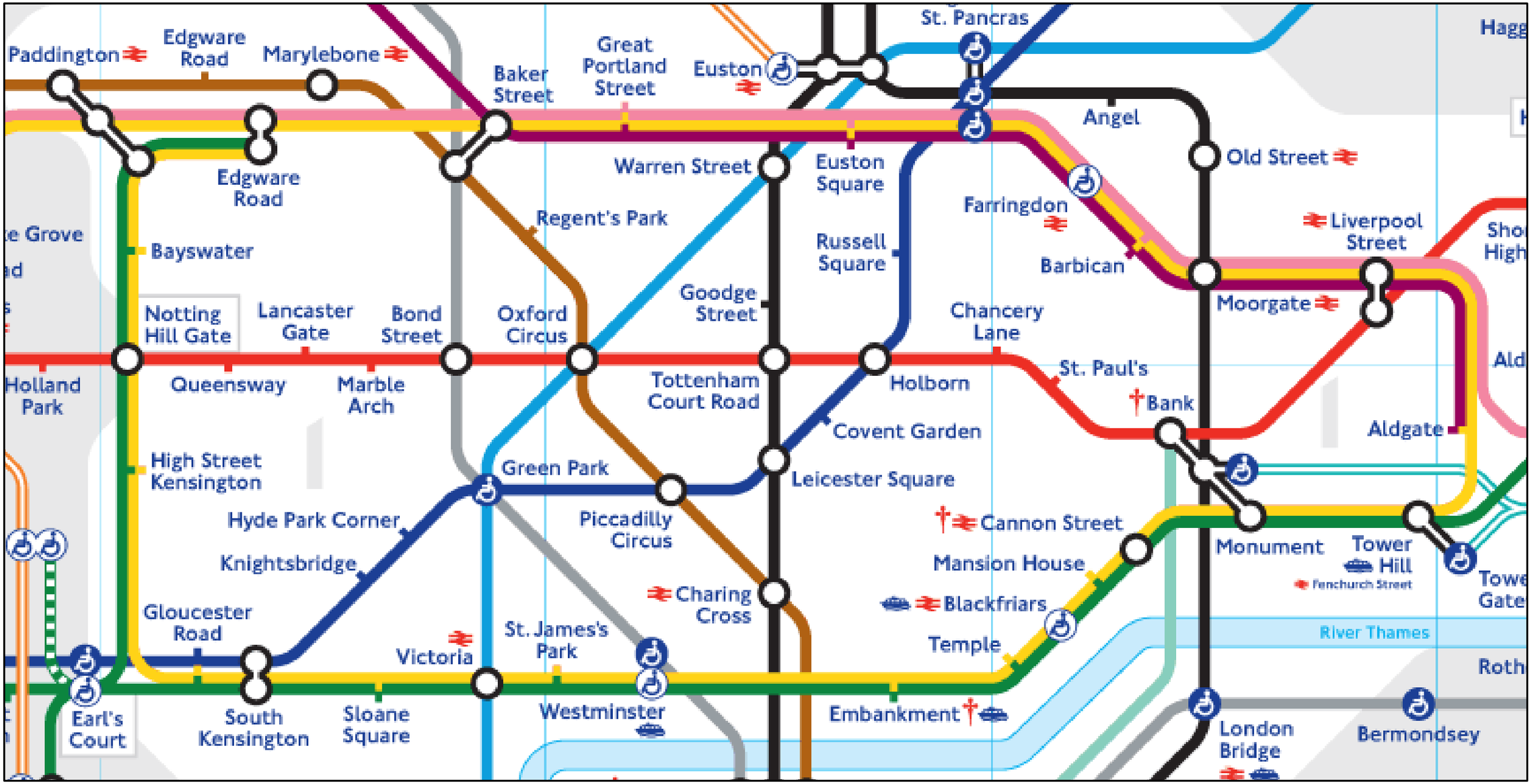


Johannes Zink

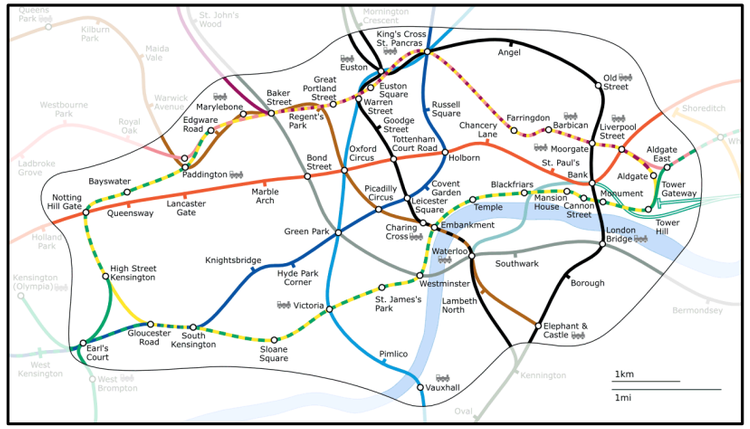
What is a Schematic Metro Map?

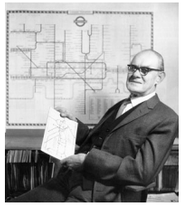


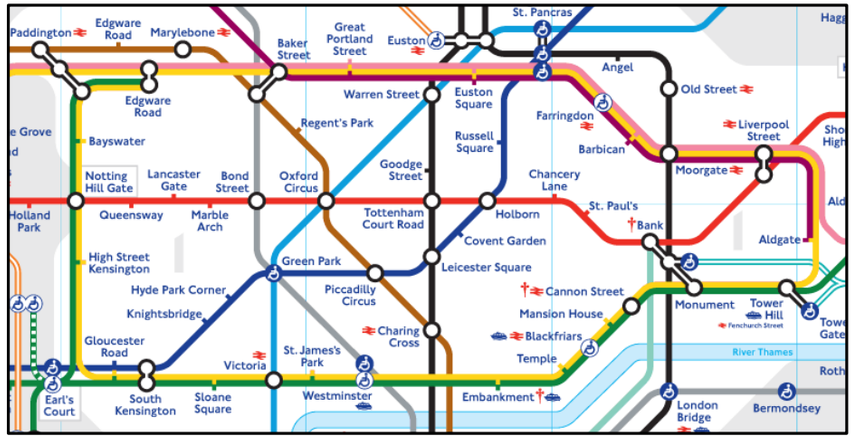
What is a Schematic Metro Map?



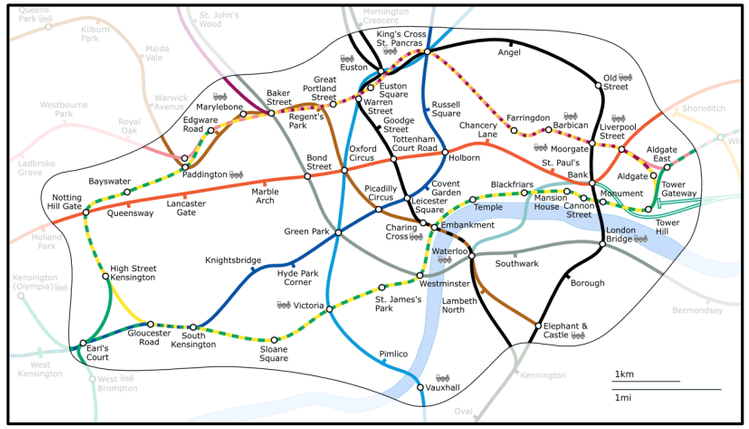
What is a Schematic Metro Map?



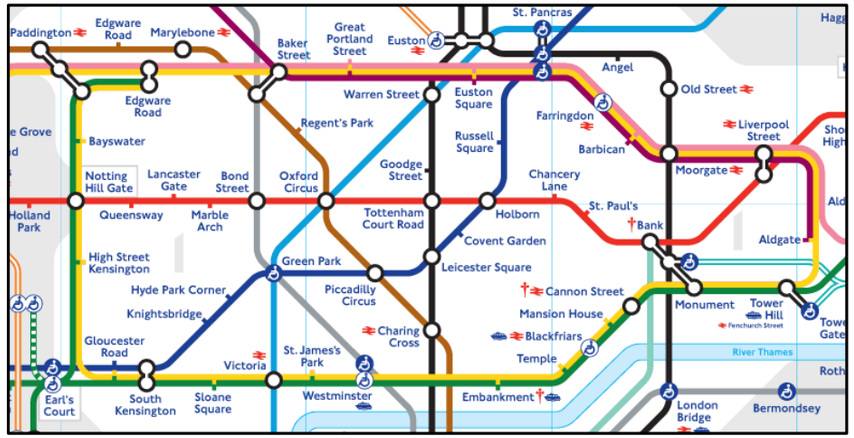




What is a Schematic Metro Map?

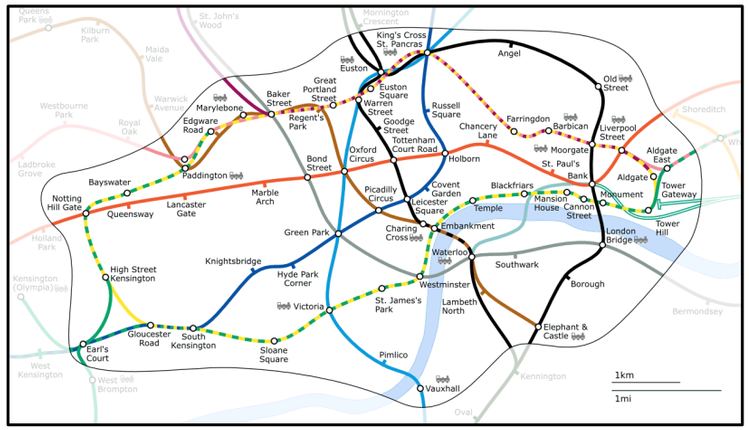






- map/diagram that shows stations connected by metro lines

What is a Schematic Metro Map?

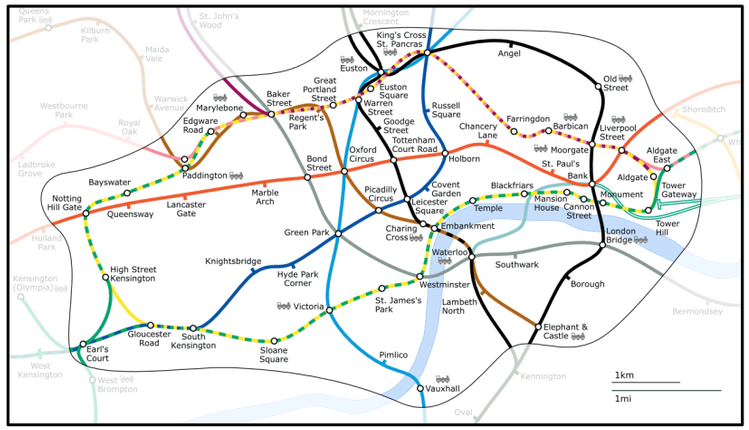






- map/diagram that shows stations connected by metro lines
- focus on topology rather than geography

What is a Schematic Metro Map?

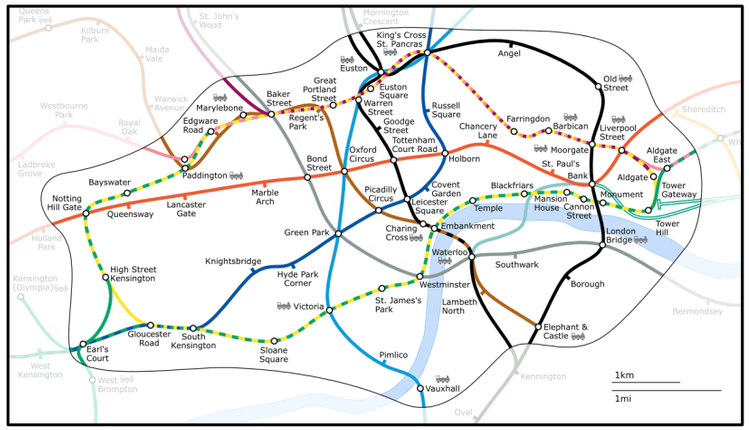






- map/diagram that shows stations connected by metro lines
- focus on topology rather than geography
- goal: easy-to-use visual navigation aid for passengers

What is a Schematic Metro Map?

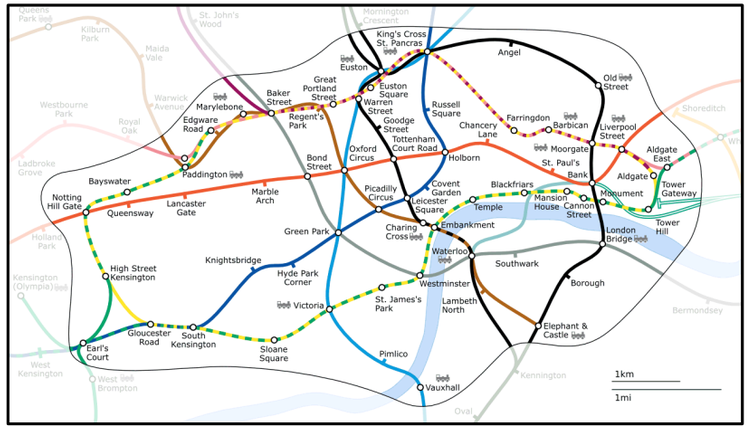




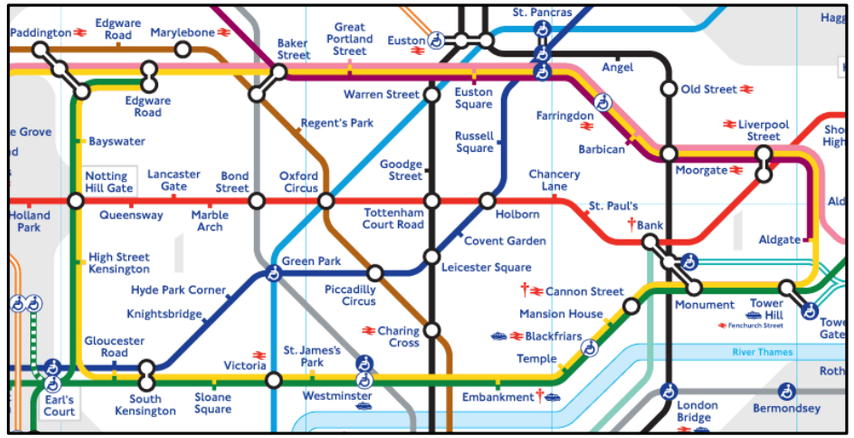


- map/diagram that shows stations connected by metro lines
- focus on topology rather than geography
- goal: easy-to-use visual navigation aid for passengers
  - "*How do I quickly get from A to B?*"

What is a Schematic Metro Map?

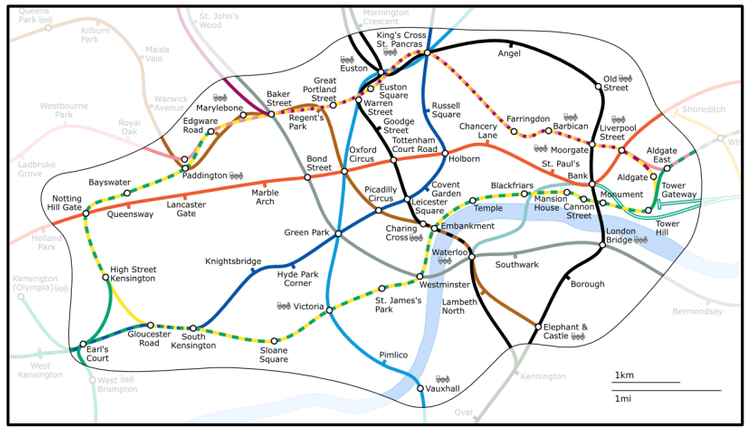






- map/diagram that shows stations connected by metro lines
- focus on topology rather than geography
- goal: easy-to-use visual navigation aid for passengers
  - "*How do I quickly get from A to B?*"
  - "*Where do I need to change trains?*"

What is a Schematic Metro Map?

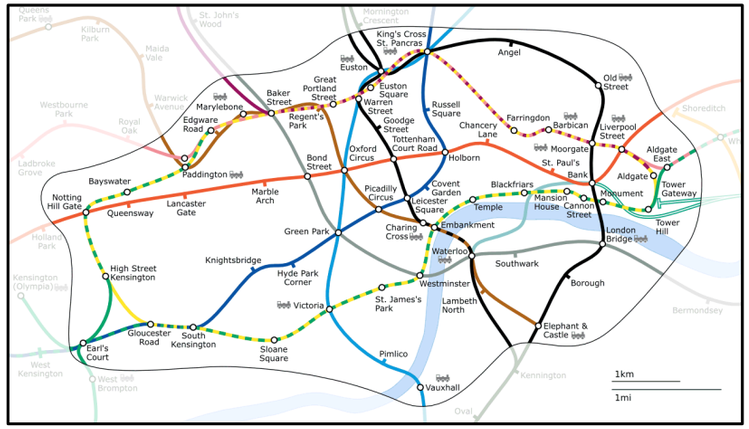




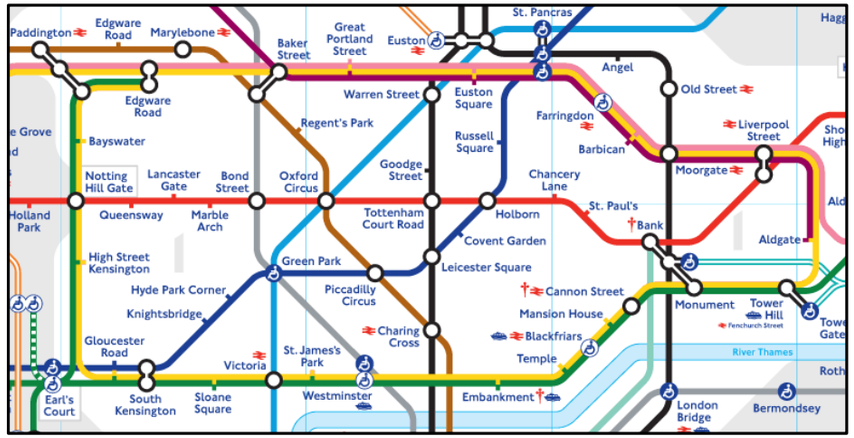


- map/diagram that shows stations connected by metro lines
- focus on topology rather than geography
- goal: easy-to-use visual navigation aid for passengers
  - "*How do I quickly get from A to B?*"
  - "*Where do I need to change trains?*"
- distorts scale and geometry

What is a Schematic Metro Map?

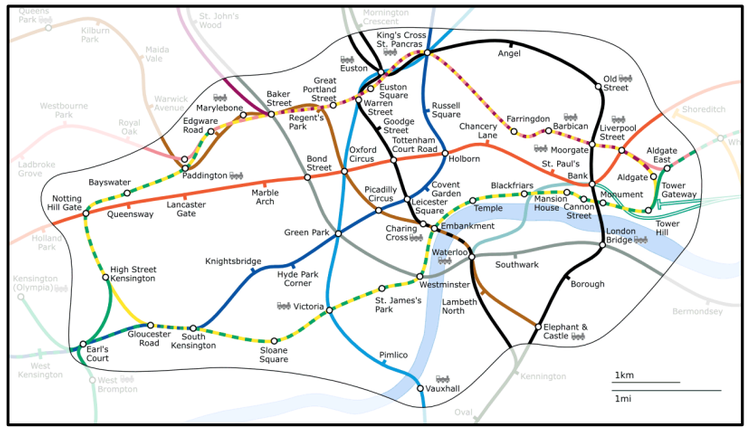






- map/diagram that shows stations connected by metro lines
- focus on topology rather than geography
- goal: easy-to-use visual navigation aid for passengers
  - "*How do I quickly get from A to B?*"
  - "*Where do I need to change trains?*"
- distorts scale and geometry
- metro map design still a largely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

What is a Schematic Metro Map?

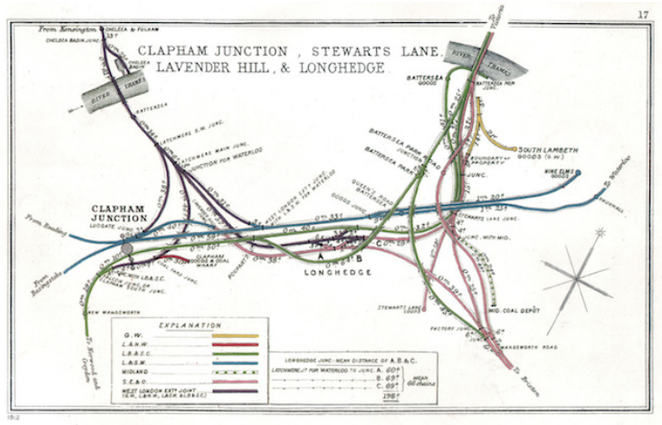






- map/diagram that shows stations connected by metro lines
- focus on topology rather than geography
- goal: easy-to-use visual navigation aid for passengers
  - "*How do I quickly get from A to B?*"
  - "*Where do I need to change trains?*"
- distorts scale and geometry
- metro map design still a largely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

Subtasks in Metro Map Layout



+



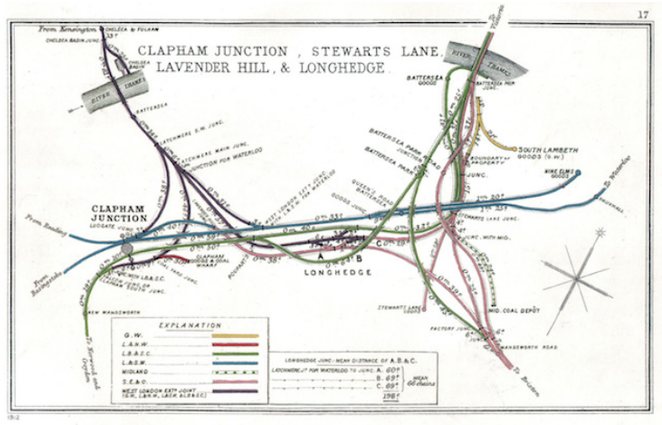
→



Input:

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

Subtasks in Metro Map Layout



+



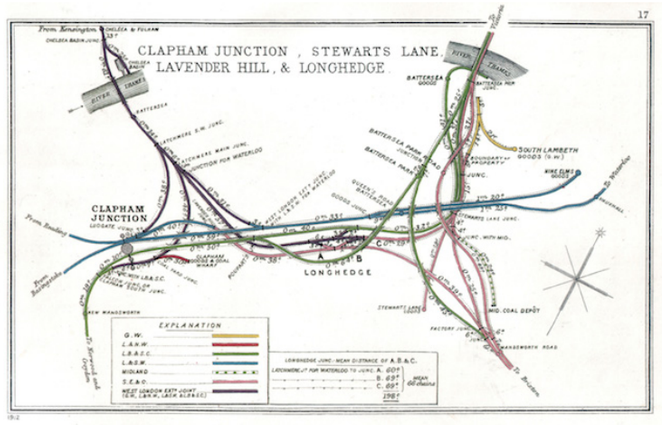
→



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



+



→

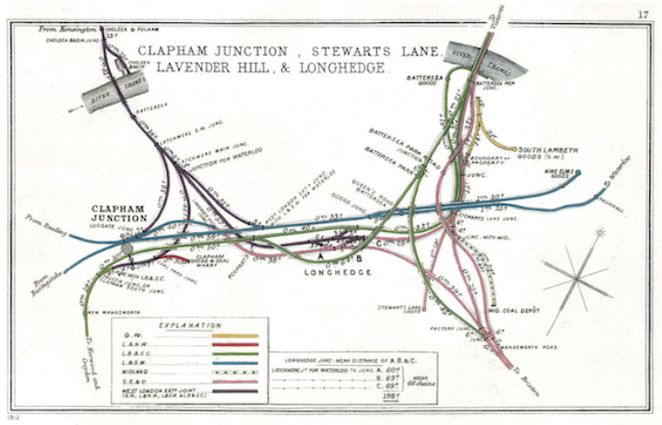


Input:
- geographically embedded railway network G
- set $\mathcal{L}$ of metro lines serving G

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

Divide the task into several subtasks:

Subtasks in Metro Map Layout



+



→

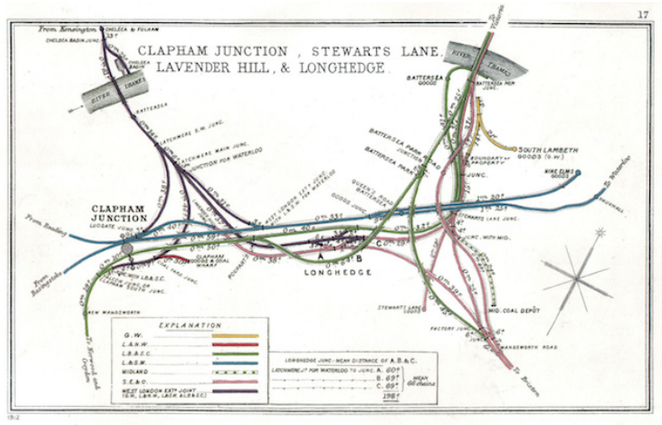


Input:
- geographically embedded railway network G
- set $\mathcal{L}$ of metro lines serving G

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

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

Subtasks in Metro Map Layout



+



→

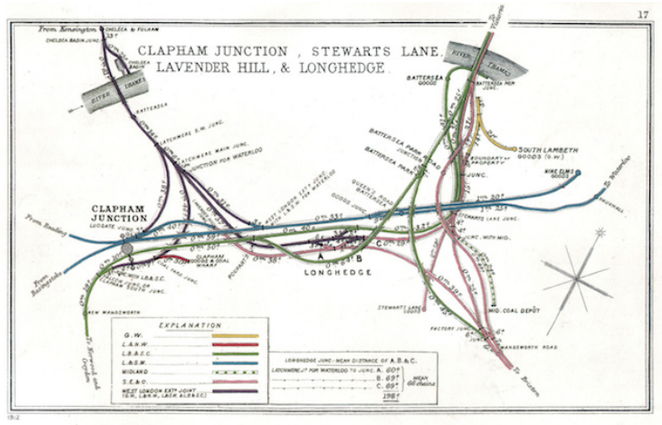


Input:
- geographically embedded railway network G
- set $\mathcal{L}$ of metro lines serving G

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



+



→

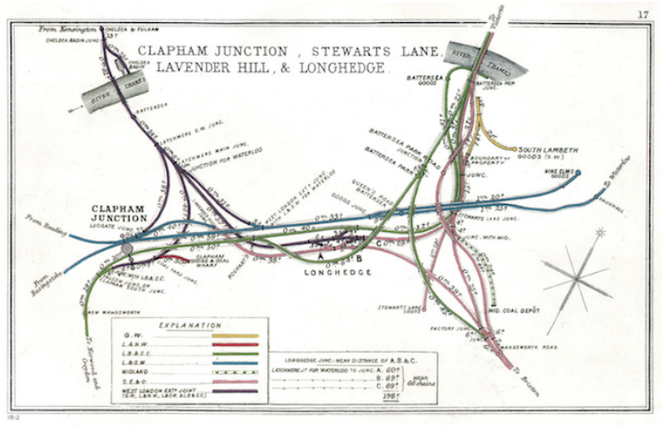


Input:
- geographically embedded railway network G
- set $\mathcal{L}$ of metro lines serving G

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



+



→

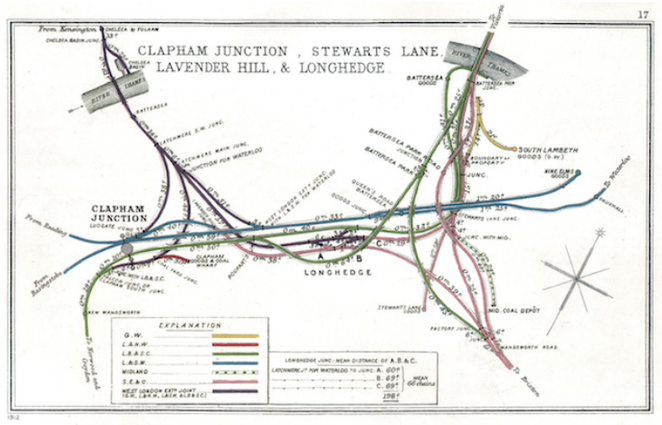


Input:
- geographically embedded railway network G
- set $\mathcal{L}$ of metro lines serving G

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



+



→



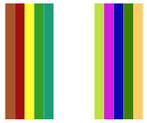
Input:
- geographically embedded railway network G
- set $\mathcal{L}$ of metro lines serving G

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

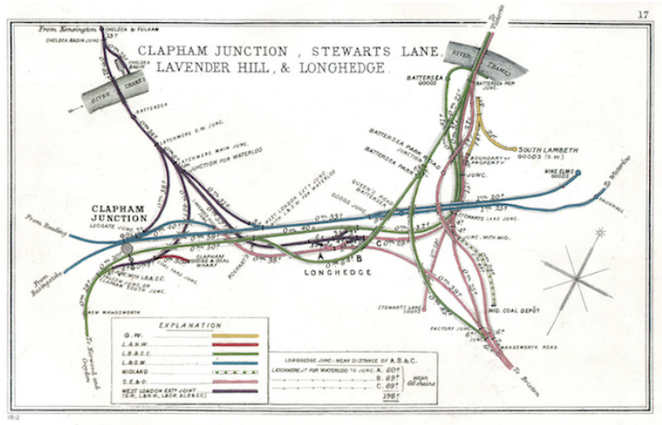
Divide the task into several subtasks:

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

colors



Subtasks in Metro Map Layout



+



→



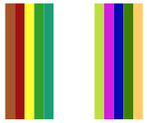
Input:
- geographically embedded railway network G
- set $\mathcal{L}$ of metro lines serving G

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

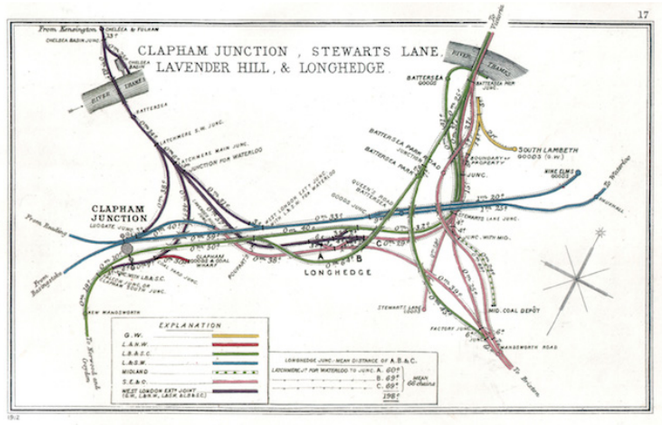
colors



fonts

Paddington
PADDINGTON
Paddington

Subtasks in Metro Map Layout



+



→



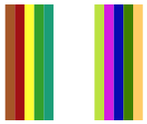
Input:
- geographically embedded railway network G
- set $\mathcal{L}$ of metro lines serving G

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

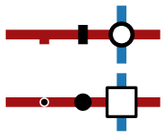
colors



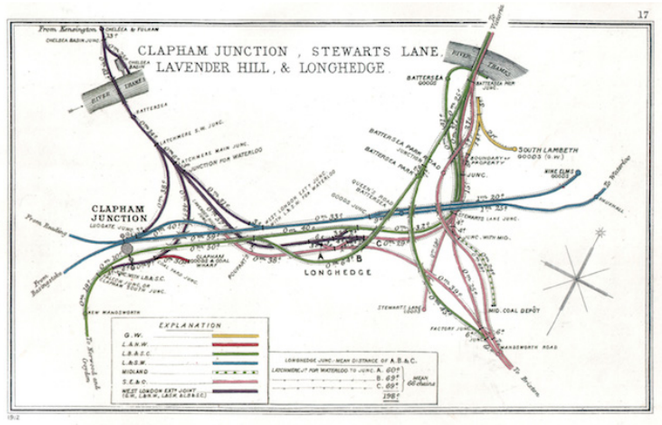
fonts

Paddington
PADDINGTON
Paddington

symbols



Subtasks in Metro Map Layout



+



→



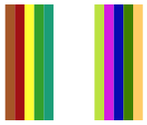
Input:
- geographically embedded railway network G
- set $\mathcal{L}$ of metro lines serving G

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

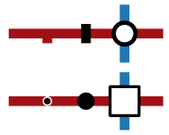
Divide the task into several subtasks:

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

colors fonts symbols

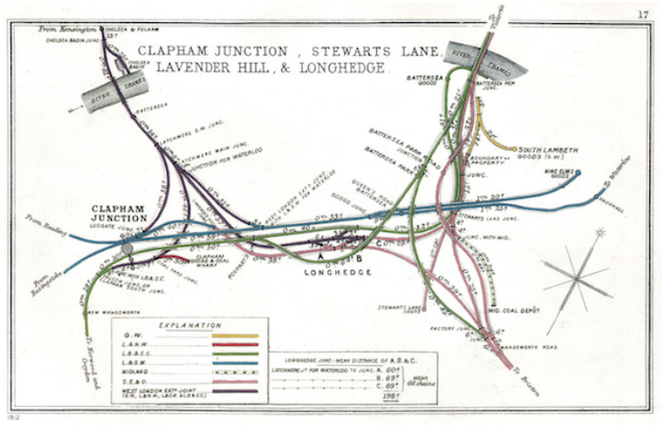


Paddington
PADDINGTON
Paddington



→ very salient,
but not a computational problem

Subtasks in Metro Map Layout



+



→

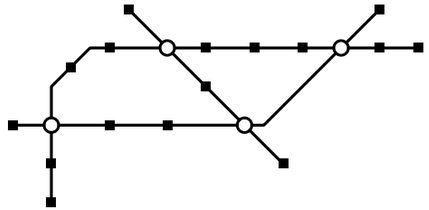


Input:
- geographically embedded railway network G
- set $\mathcal{L}$ of metro lines serving G

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

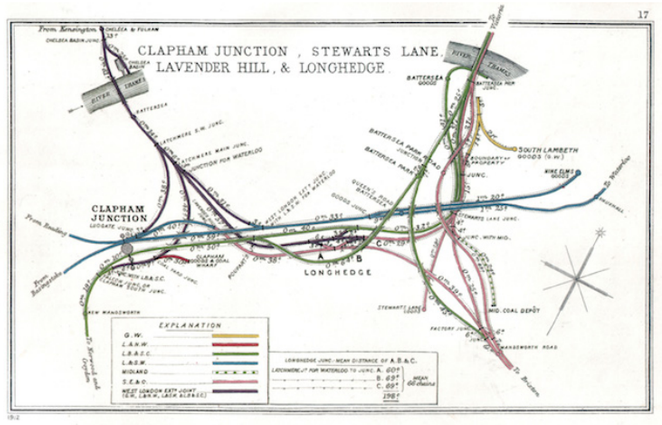
Divide the task into several subtasks:

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



determine geometry of network layout

Subtasks in Metro Map Layout



+



→

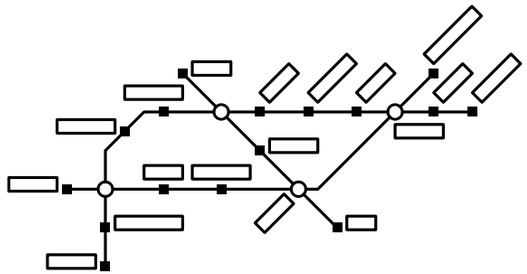


Input:
- geographically embedded railway network G
- set $\mathcal{L}$ of metro lines serving G

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

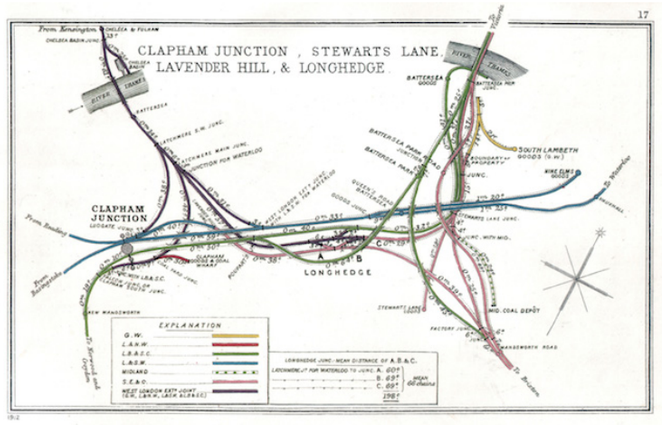
Divide the task into several subtasks:

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



determine positions of station names

Subtasks in Metro Map Layout



+



→

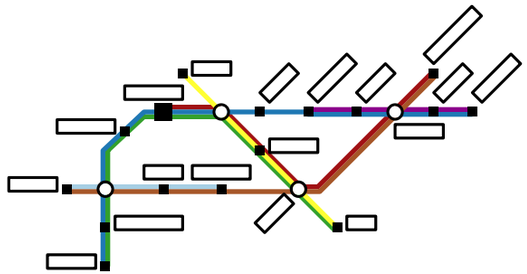


Input:
- geographically embedded railway network G
- set $\mathcal{L}$ of metro lines serving G

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

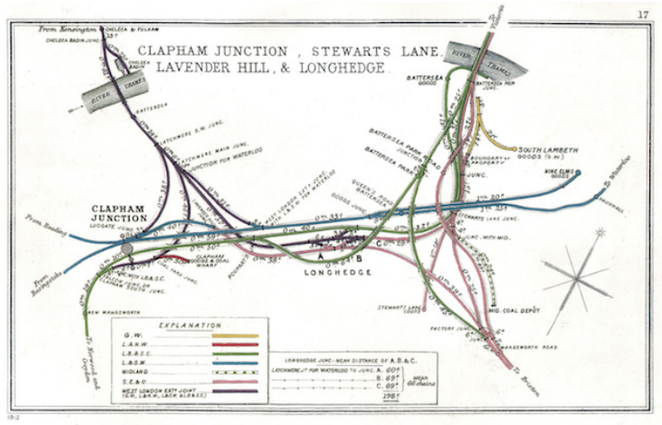
Divide the task into several subtasks:

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



determine line routing and ordering of bundles

Subtasks in Metro Map Layout



+



→



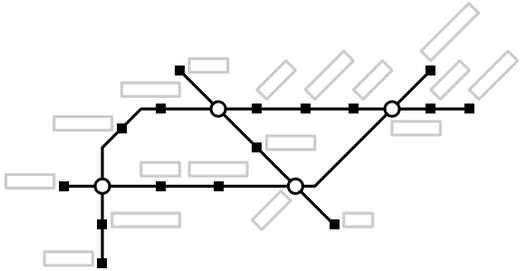
Input:
- geographically embedded railway network G
- set $\mathcal{L}$ of metro lines serving G

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?

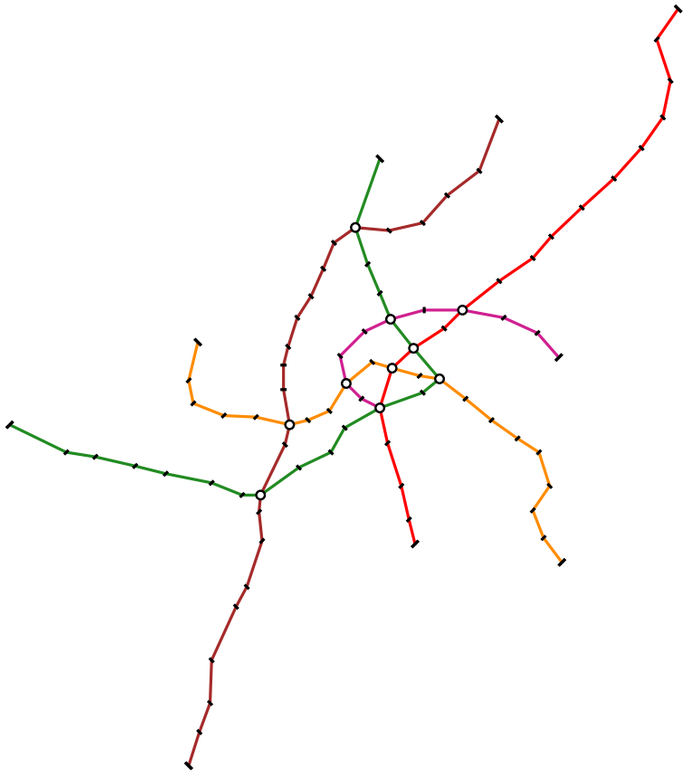
$\rightarrow$ extract common principles of existing, manually designed metro maps

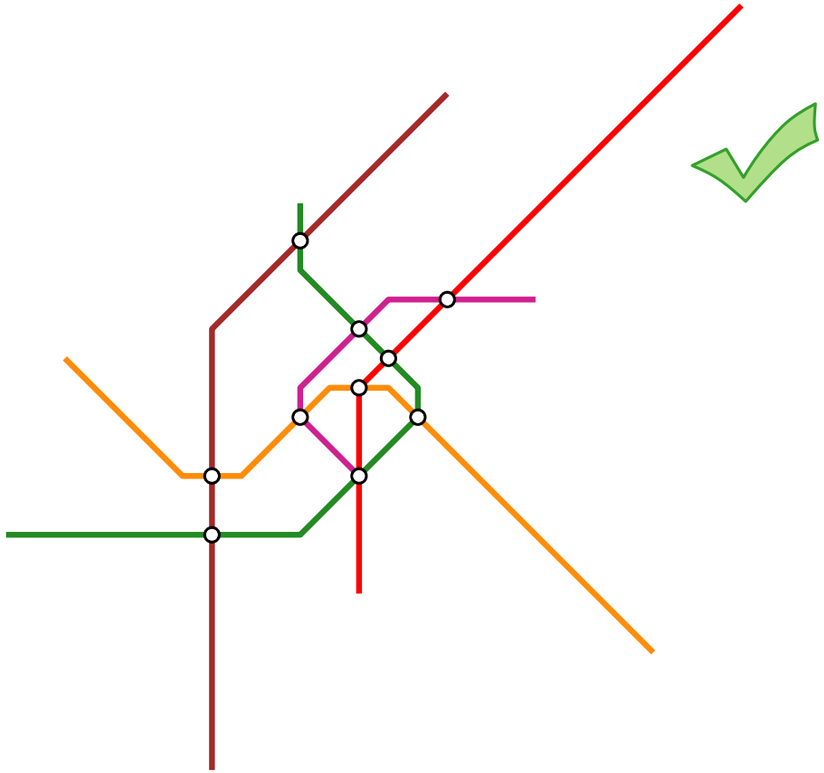


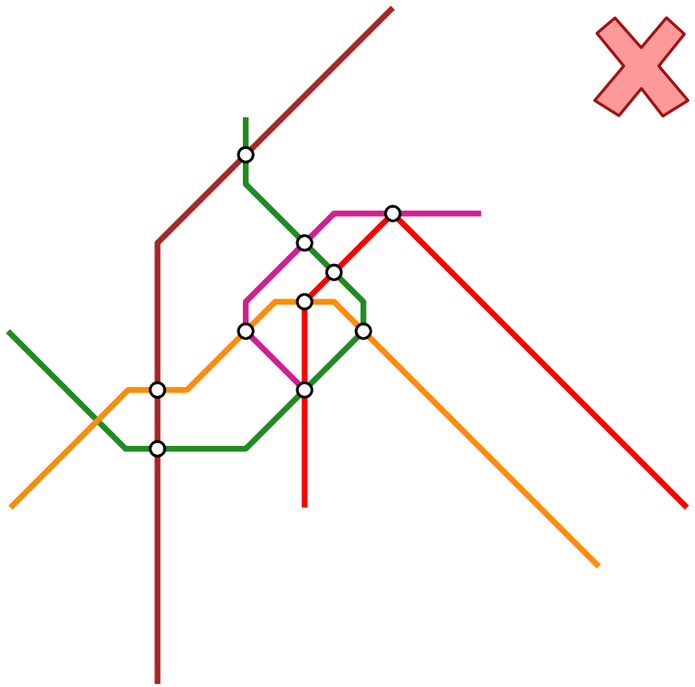
Design Rules

(R1) Do not change the network topology.

- no new crossings
- no changes in circular vertex orders





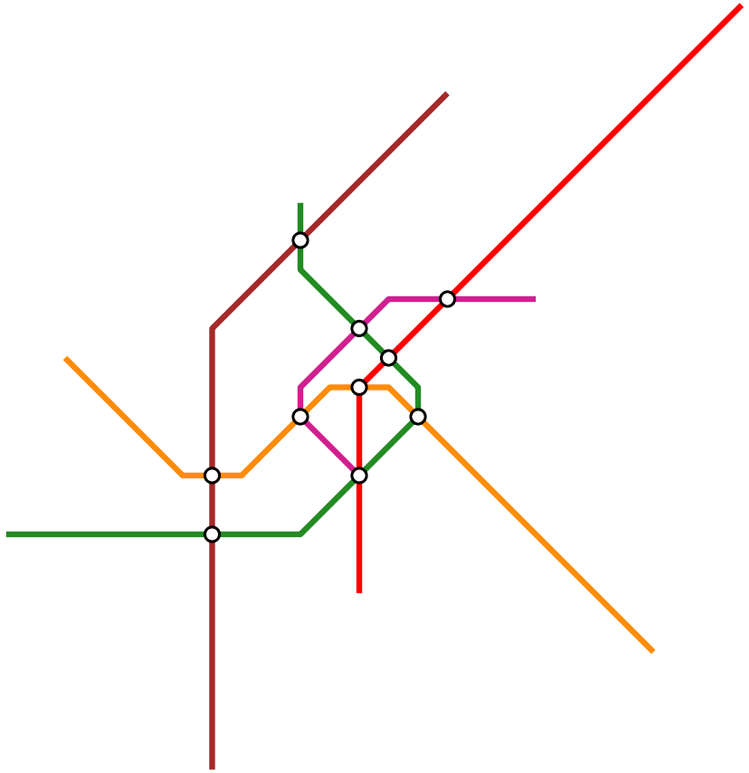


Design Rules

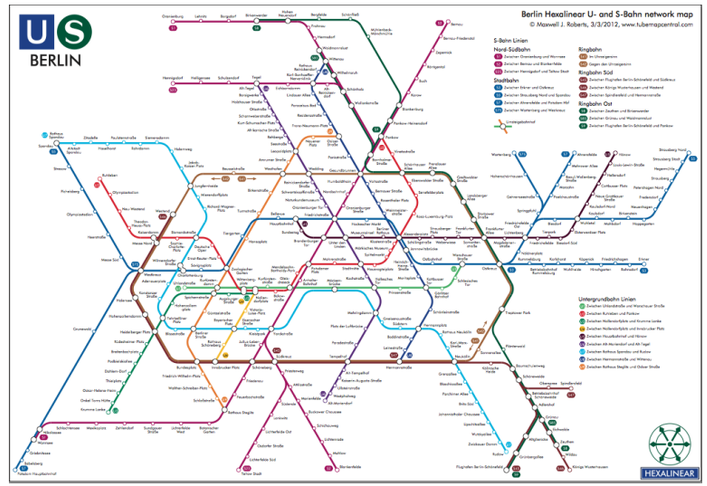
(R1) Do not change the network topology.

(R2) Restrict edge orientations.

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







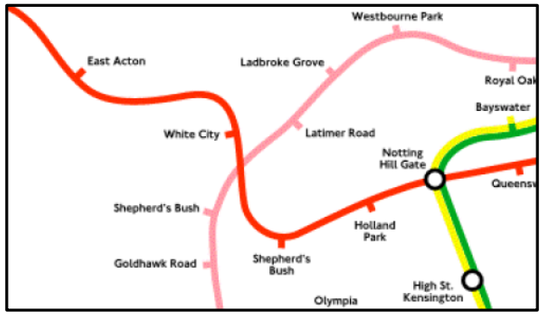
Design Rules

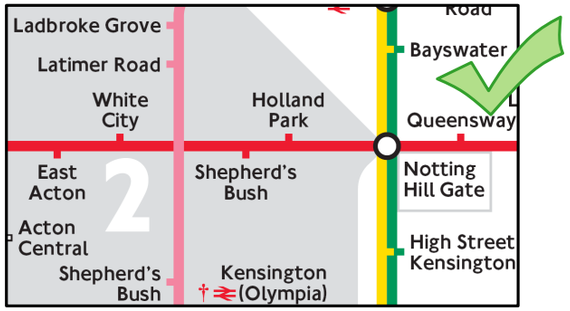
(R1) Do not change the network topology.

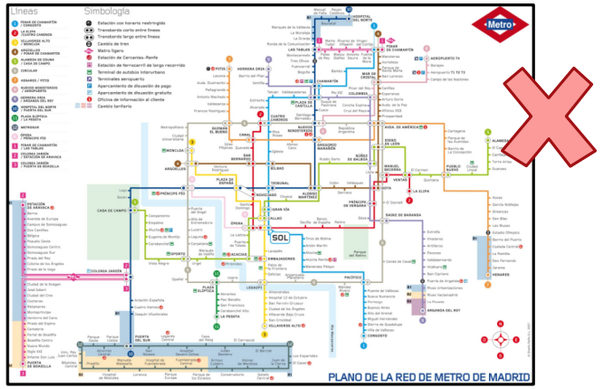
(R2) Restrict edge orientations.

(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.

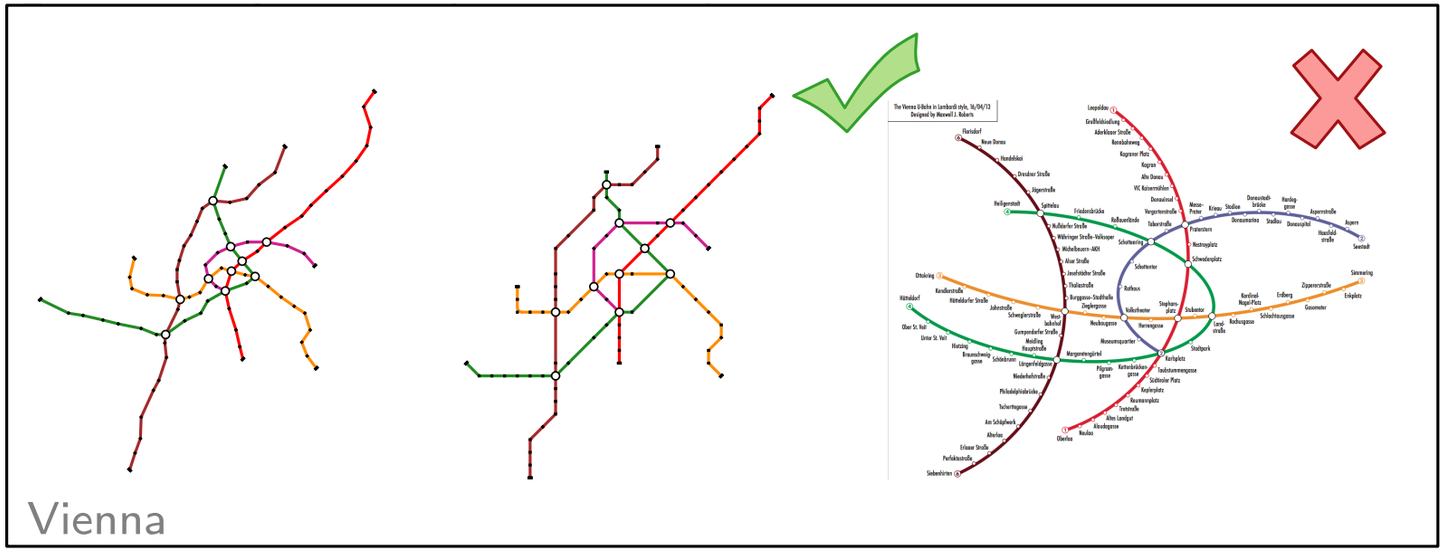
(R5) Use large angular resolution in stations.

- 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.

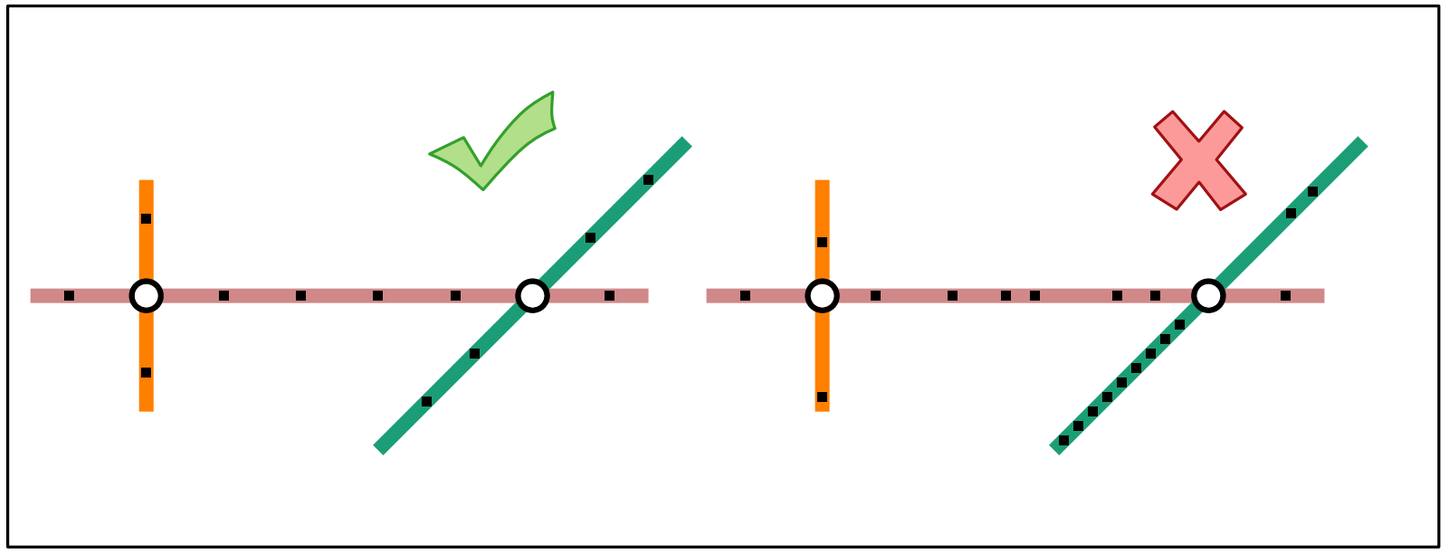
(R2)

(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.

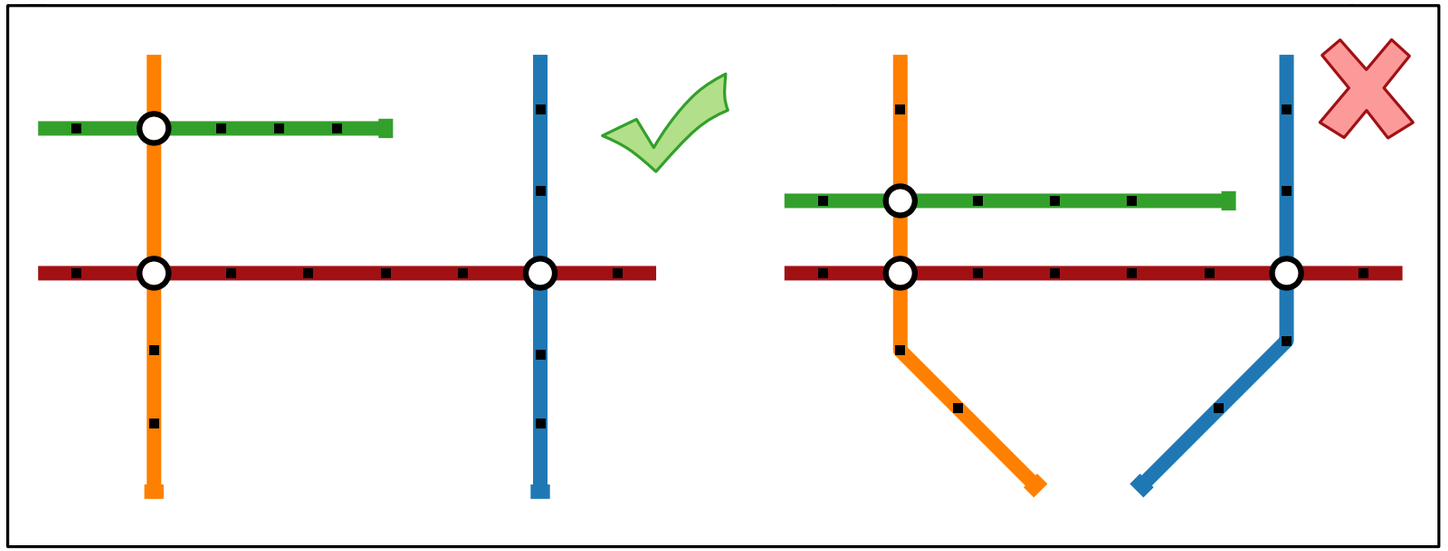
(R3)

(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)

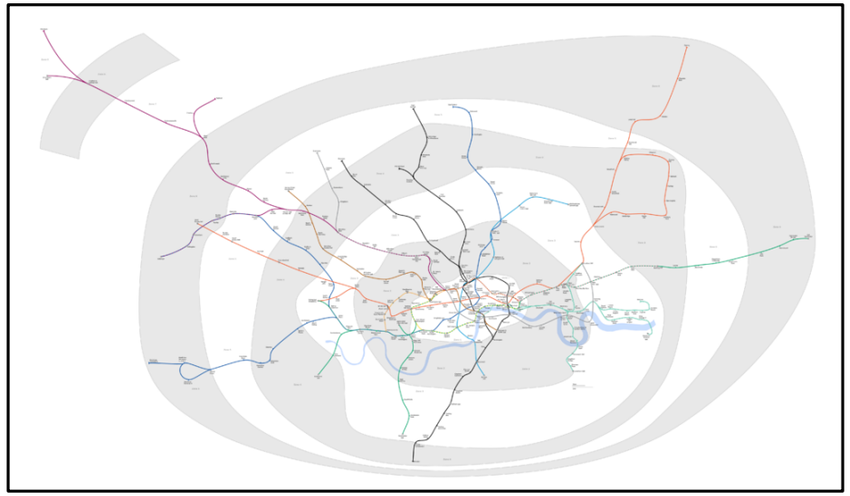
(R4)

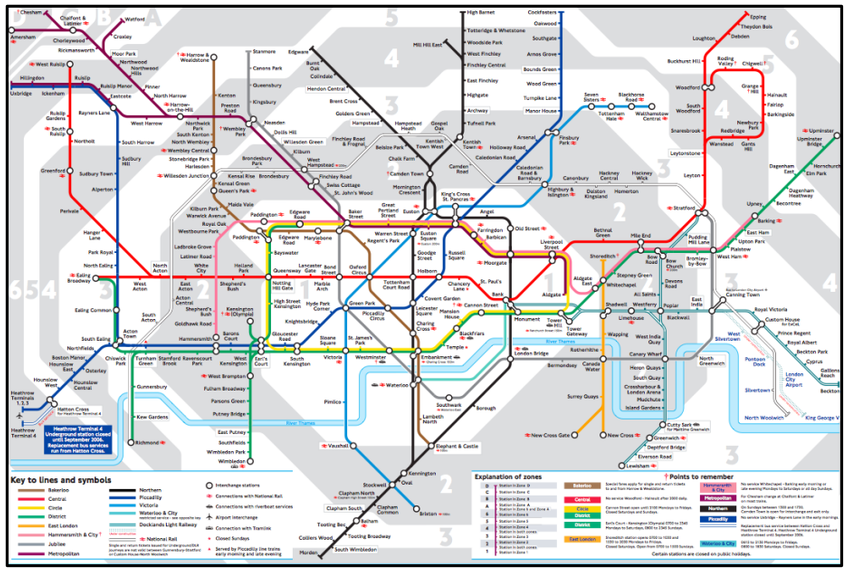
(R5)

(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.

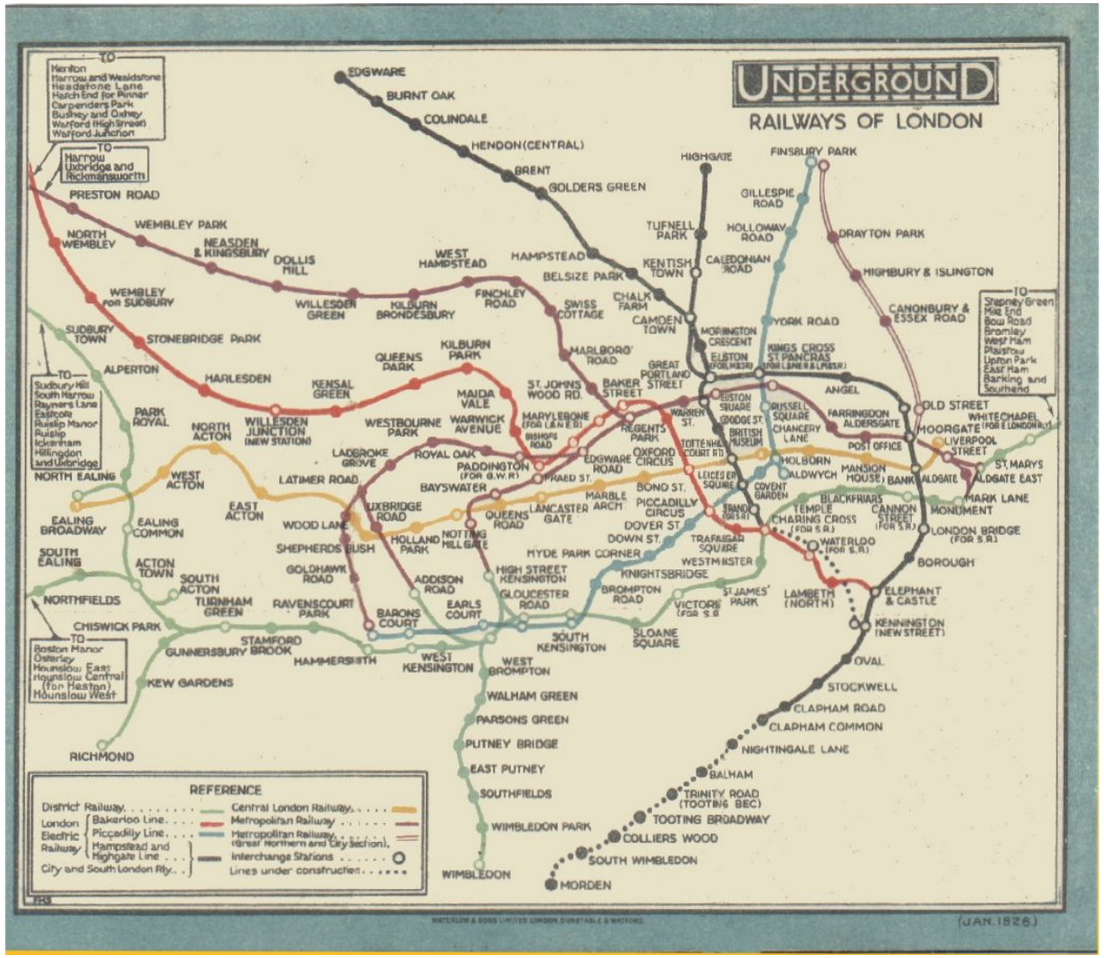
(R7) Use uniform edge lengths.

(R8) Keep unrelated features apart.

(R9) Avoid large empty spaces.

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

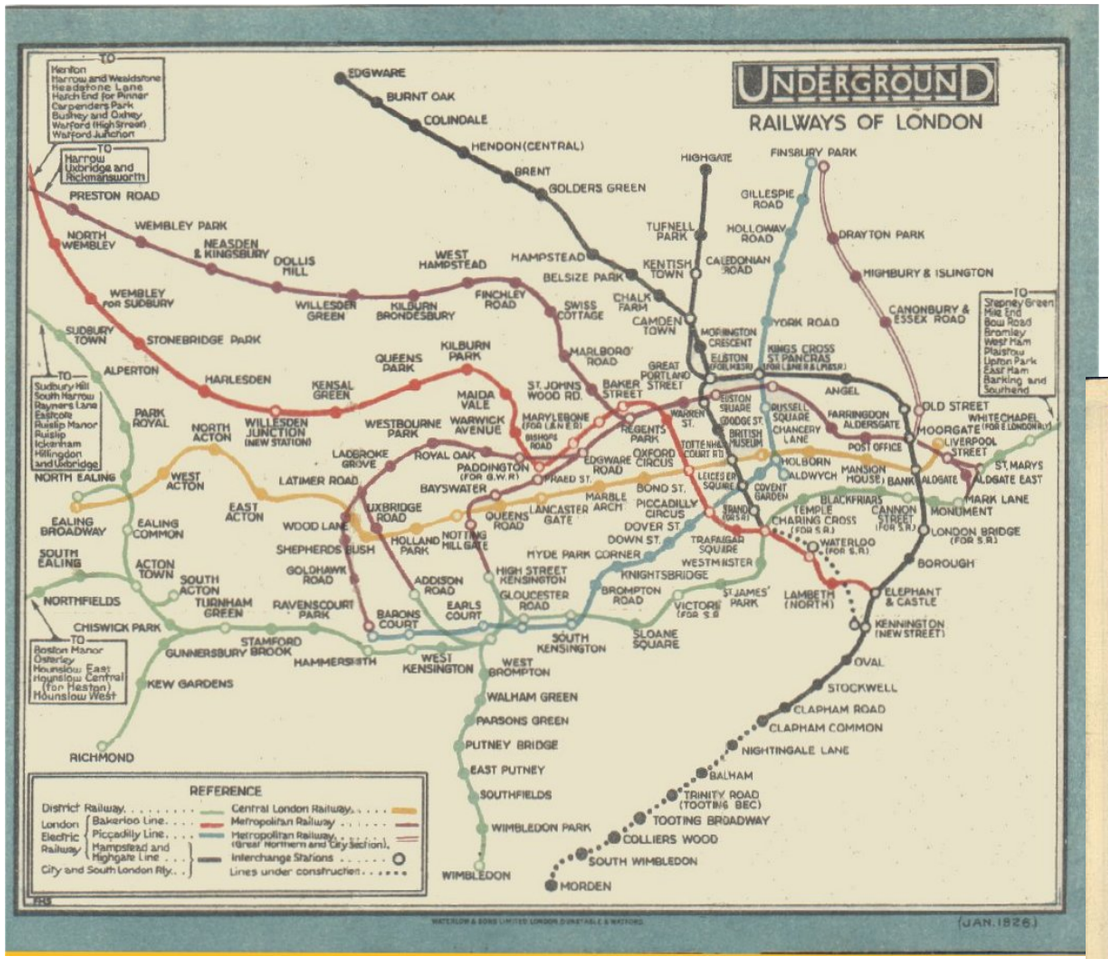
A Bit of History



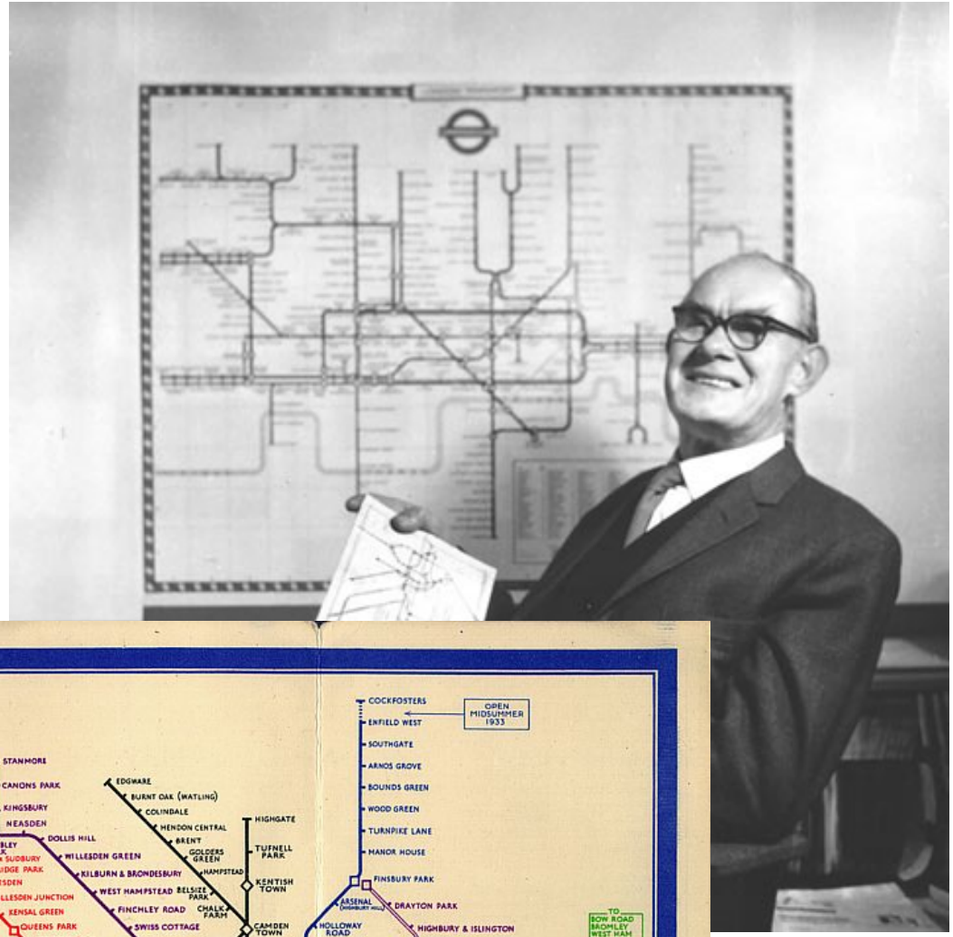
London 1927 (Fred H. Stingemore)

(c) Mike Yashworth

A Bit of History



London 1927 (Fred H. Stingemore)

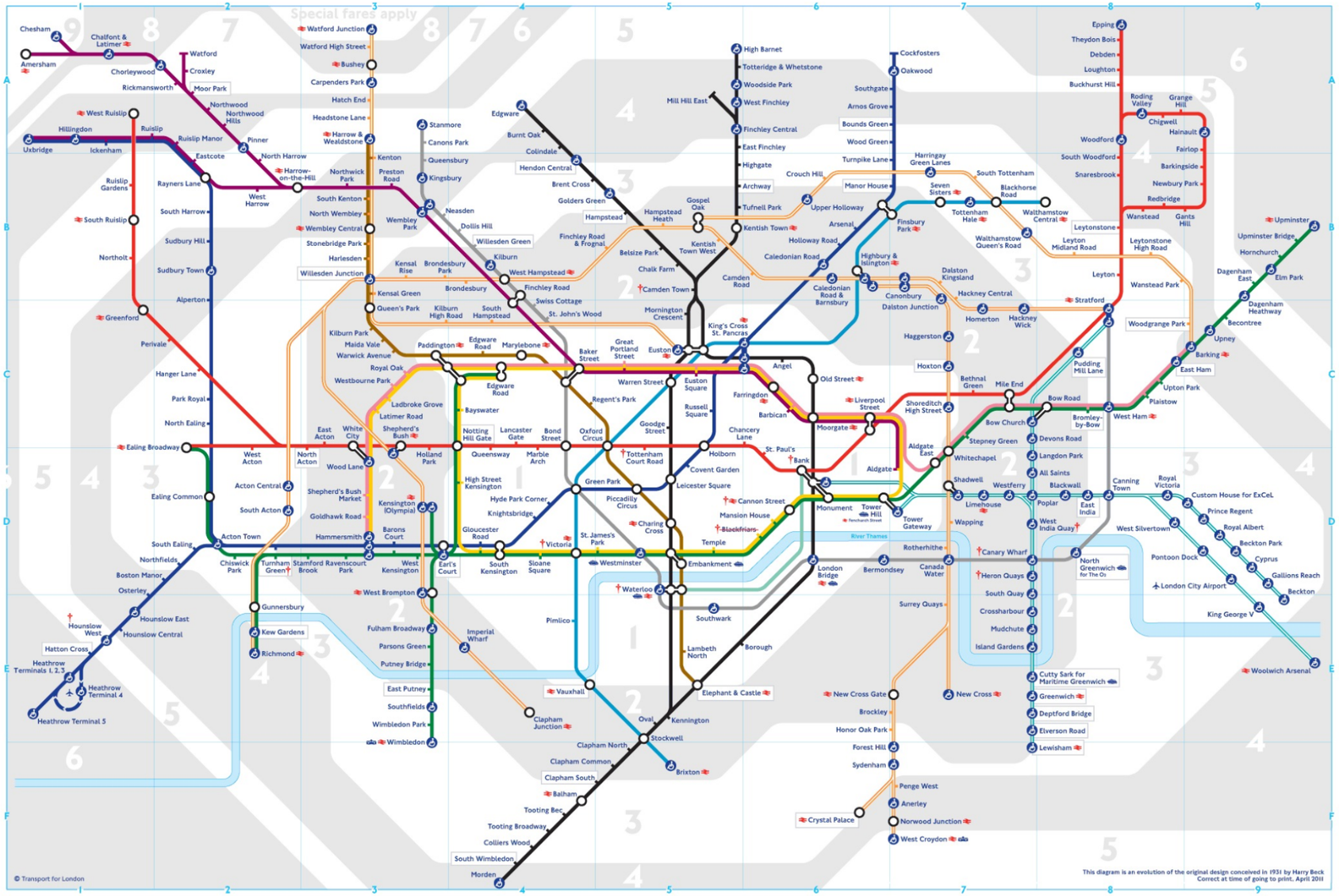


Henry Beck
London 1933



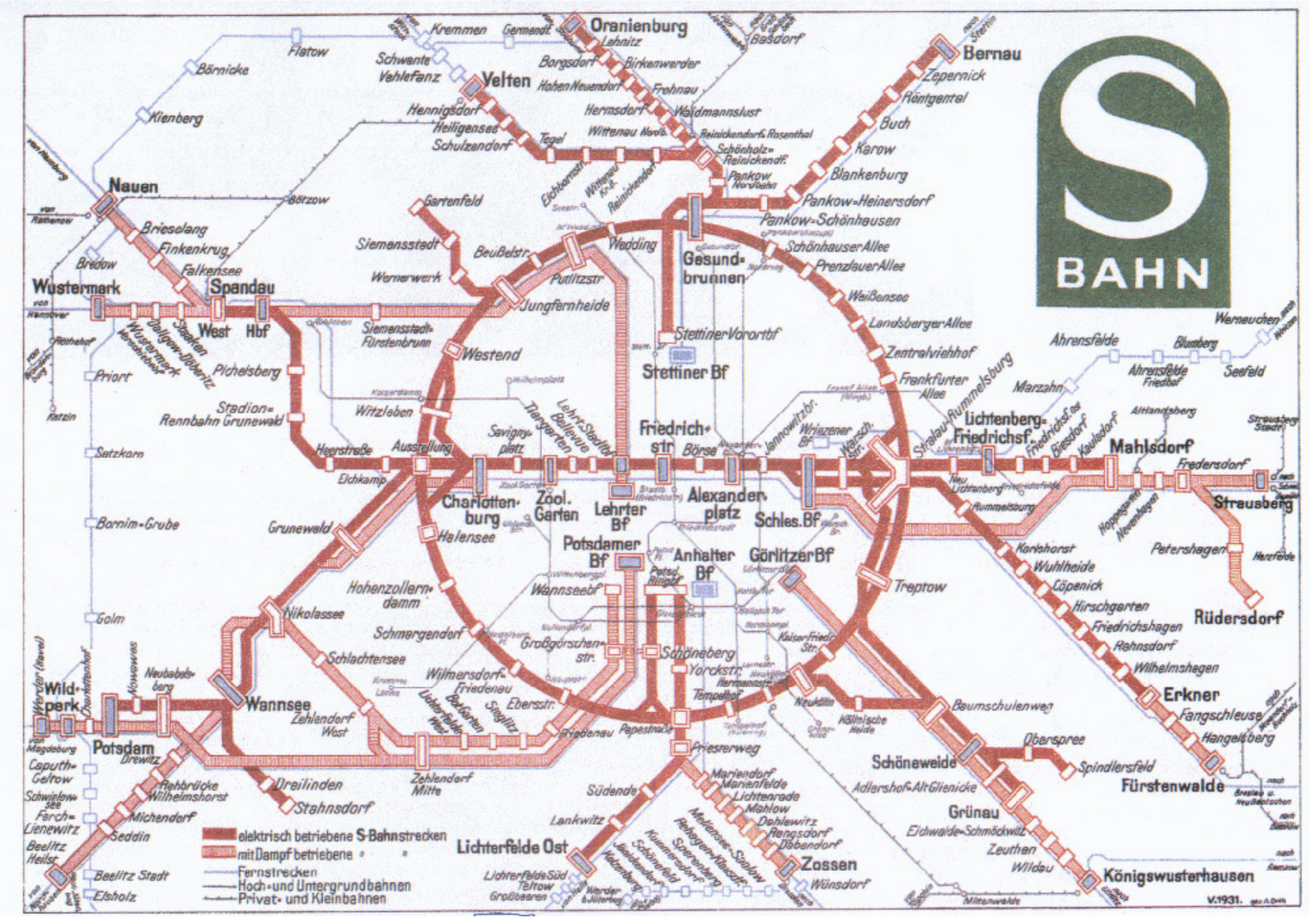
(c) Mike Yashworth

A Bit of History

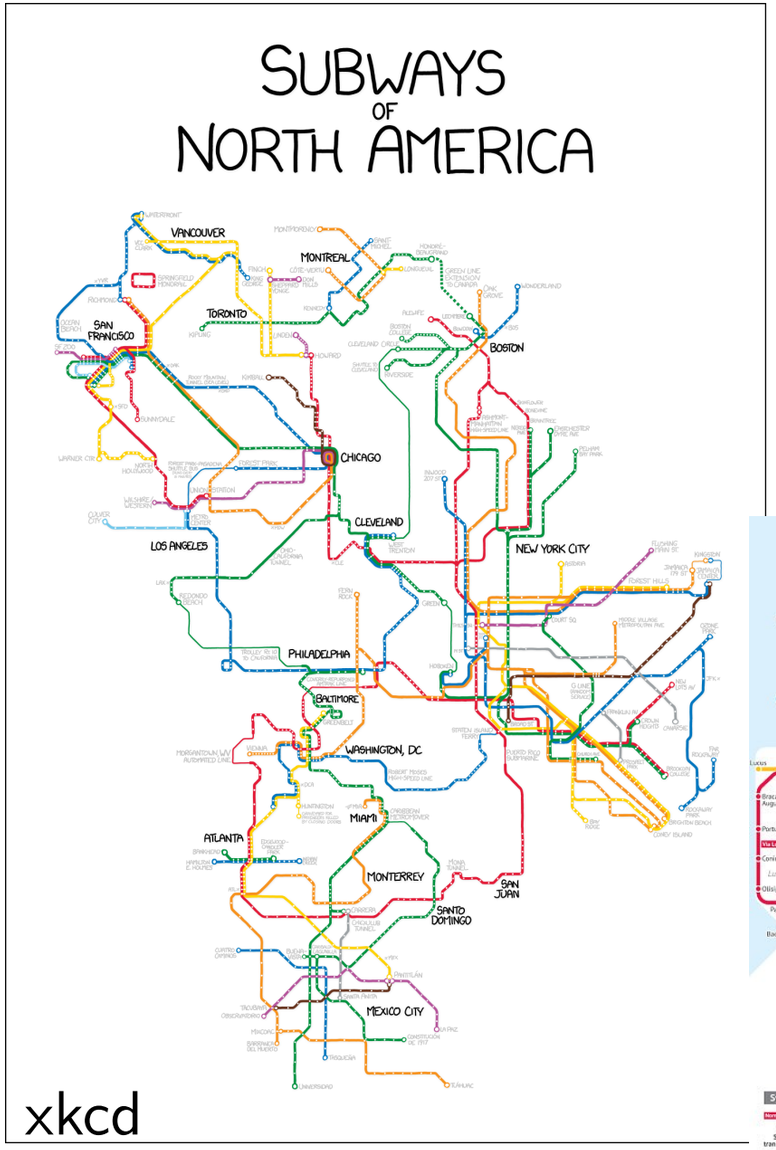

Special fares apply
Chesham
Amersham
Chalfont & Latimer
Watford
Croxley
Chorleywood
Rickmansworth
Moor Park
Northwood
Northwood Hills
Pinner
North Harrow
Harrow-on-the-Hill
West Harrow
Uxbridge
Hillingdon
Ickenham
Ruislip
Ruislip Manor
Eastcote
Ruislip Gardens
Rayners Lane
South Ruislip
South Harrow
Sudbury Hill
Sudbury Town
Northolt
Greenford
Perivale
Hanger Lane
Park Royal
North Ealing
Ealing Broadway
West Acton
North Acton
East Acton
White City
Wood Lane
Shepherd's Bush
Latimer Road
Ladbroke Grove
Westbourne Park
Royal Oak
Warwick Avenue
Maida Vale
Kilburn Park
Queen's Park
Kensal Green
Willesden Junction
Harlesden
Stonebridge Park
Wembley Central
North Wembley
South Kenton
Kenton
Northwick Park
Preston Road
Wembley Park
Neasden
Dollis Hill
Willesden Green
Kilburn
West Hampstead
Kilburn High Road
South Hampstead
Finchley Road
Swiss Cottage
St. John's Wood
Brondesbury
Brondesbury Park
Kensal Rise
Acton Central
South Acton
Acton Town
Ealing Common
Shepherd's Bush Market
Goldhawk Road
Kensington (Olympia)
Barons Court
Hammersmith
South Ealing
Northfields
Boston Manor
Osterley
Hounslow West
Hounslow East
Hounslow Central
Hatton Cross
Heathrow Terminals 1, 2, 3
Heathrow Terminal 4
Heathrow Terminal 5
Chiswick Park
Turnham Green
Stamford Brook
Ravenscourt Park
West Kensington
Earl's Court
Gunnersbury
Kew Gardens
Richmond
West Brompton
Fulham Broadway
Parsons Green
Putney Bridge
East Putney
Southfields
Wimbledon Park
Wimbledon
Imperial Wharf
Clapham Junction
Holland Park
Notting Hill Gate
Queensway
Bayswater
Lancaster Gate
Marble Arch
Bond Street
High Street Kensington
Hyde Park Corner
Knightsbridge
South Kensington
Gloucester Road
Victoria
Sloane Square
St. James's Park
Pimlico
Vauxhall
Oval
Kennington
Clapham North
Clapham Common
Clapham South
Balham
Tooting Bec
Tooting Broadway
Colliers Wood
South Wimbledon
Morden
Oxford Circus
Tottenham Court Road
Green Park
Piccadilly Circus
Charing Cross
Westminster
Waterloo
Embankment
Temple
Blackfriars
Mansion House
Cannon Street
Holborn
Covent Garden
Leicester Square
Goodge Street
Warren Street
Regent's Park
Euston
Euston Square
Great Portland Street
Baker Street
Edgware Road
Paddington
Marylebone
Russell Square
Chancery Lane
St. Paul's
Bank
Monument
Tower Hill
Aldgate
Aldgate East
Liverpool Street
Moorgate
Old Street
Angel
Farringdon
Barbican
King's Cross St. Pancras
Mornington Crescent
Camden Town
Chalk Farm
Belsize Park
Hampstead Heath
Gospel Oak
Kentish Town West
Kentish Town
Camden Road
Caledonian Road
Caledonian Road & Barnsbury
Holloway Road
Arsenal
Finsbury Park
Highbury & Islington
Canonbury
Dalston Junction
Dalston Kingsland
Hackney Central
Homerton
Hackney Wick
Haggerston
Hoxton
Shoreditch High Street
Bethnal Green
Mile End
Bow Road
Bow Church
Devons Road
Stepney Green
Whitechapel
Langdon Park
All Saints
Shadwell
Limehouse
Westferry
Poplar
Blackwall
East India
Canning Town
Royal Victoria
Custom House for ExCeL
Prince Regent
Royal Albert
Beckton Park
Cyprus
Gallions Reach
Beckton
London City Airport
King George V
Pontoon Dock
West Silvertown
Wapping
West India Quay
Canary Wharf
Heron Quays
South Quay
Crossharbour
Mudchute
Island Gardens
Cutty Sark for Maritime Greenwich
Greenwich
Deptford Bridge
Elverson Road
Lewisham
Woolwich Arsenal
Rotherhithe
Canada Water
Surrey Quays
North Greenwich for The O2
Bermondsey
London Bridge
Borough
Southwark
Lambeth North
Elephant & Castle
New Cross Gate
New Cross
Brockley
Honor Oak Park
Forest Hill
Sydenham
Penge West
Anerley
Crystal Palace
Norwood Junction
West Croydon
High Barnet
Totteridge & Whetstone
Woodside Park
West Finchley
Finchley Central
East Finchley
Highgate
Archway
Tufnell Park
Mill Hill East
Cockfosters
Oakwood
Southgate
Arnos Grove
Bounds Green
Wood Green
Turnpike Lane
Manor House
Crouch Hill
Upper Holloway
Harringay Green Lanes
Seven Sisters
South Tottenham
Blackhorse Road
Tottenham Hale
Walthamstow Central
Walthamstow Queen's Road
Leytonstone
Leyton Midland Road
Leytonstone High Road
Leyton
Wanstead Park
Wanstead
Gants Hill
Redbridge
Newbury Park
Barkingside
Fairlop
Hainault
Chigwell
Roding Valley
Grange Hill
Epping
Theydon Bois
Debden
Loughton
Buckhurst Hill
Woodford
South Woodford
Snaresbrook
Stratford
Pudding Mill Lane
Woodgrange Park
Upton Park
East Ham
Barking
Upney
Becontree
Dagenham Heathway
Dagenham East
Elm Park
Hornchurch
Upminster Bridge
Upminster
Plaistow
West Ham
Bromley-by-Bow
Edgware
Burnt Oak
Colindale
Hendon Central
Stanmore
Canons Park
Queensbury
Kingsbury
Harrow & Wealdstone
Hatch End
Headstone Lane
Carpenders Park
Bushey
Watford High Street
Watford Junction
Mill Hill East
Edgware
Burnt Oak
Colindale
Hendon Central
Brent Cross
Golders Green
Hampstead
Finchley Road & Frognal
Swiss Cottage
St. John's Wood
Mornington Crescent
Camden Town
Chalk Farm
Belsize Park
Hampstead Heath
Gospel Oak
Kentish Town West
Kentish Town
Camden Road
Caledonian Road
Caledonian Road & Barnsbury
Holloway Road
Arsenal
Finsbury Park
Highbury & Islington
Canonbury
Dalston Junction
Dalston Kingsland
Hackney Central
Homerton
Hackney Wick
Haggerston
Hoxton
Shoreditch High Street
Bethnal Green
Mile End
Bow Road
Bow Church
Devons Road
Stepney Green
Whitechapel
Langdon Park
All Saints
Shadwell
Limehouse
Westferry
Poplar
Blackwall
East India
Canning Town
Royal Victoria
Custom House for ExCeL
Prince Regent
Royal Albert
Beckton Park
Cyprus
Gallions Reach
Beckton
London City Airport
King George V
Pontoon Dock
West Silvertown
Wapping
West India Quay
Canary Wharf
Heron Quays
South Quay
Crossharbour
Mudchute
Island Gardens
Cutty Sark for Maritime Greenwich
Greenwich
Deptford Bridge
Elverson Road
Lewisham
Woolwich Arsenal
Rotherhithe
Canada Water
Surrey Quays
North Greenwich for The O2
Bermondsey
London Bridge
Borough
Southwark
Lambeth North
Elephant & Castle
New Cross Gate
New Cross
Brockley
Honor Oak Park
Forest Hill
Sydenham
Penge West
Anerley
Crystal Palace
Norwood Junction
West Croydon
Epping
Theydon Bois
Debden
Loughton
Buckhurst Hill
Roding Valley
Grange Hill
Chigwell
Hainault
Woodford
South Woodford
Snaresbrook
Fairlop
Barkingside
Newbury Park
Redbridge
Wanstead
Gants Hill
Leytonstone
Leyton Midland Road
Leytonstone High Road
Leyton
Wanstead Park
Woodgrange Park
Upminster
Upminster Bridge
Hornchurch
Dagenham East
Elm Park
Dagenham Heathway
Becontree
Upney
Barking
East Ham
Upton Park
Plaistow
West Ham
Bromley-by-Bow
Pudding Mill Lane
Stratford
Wembley Park
Wembley Central
Stonebridge Park
Harlesden
Willesden Junction
Kensal Rise
Brondesbury Park
Brondesbury
Kilburn High Road
Kilburn
West Hampstead
Finchley Road
Swiss Cottage
St. John's Wood
Queen's Park
Kensal Green
Willesden Junction
Harlesden
Stonebridge Park
Wembley Central
North Wembley
South Kenton
Kenton
Northwick Park
Preston Road
Wembley Park
Neasden
Dollis Hill
Willesden Green
Kilburn
West Hampstead
Kilburn High Road
South Hampstead
Finchley Road
Swiss Cottage
St. John's Wood
Mornington Crescent
King's Cross St. Pancras
Euston
Great Portland Street
Baker Street
Edgware Road
Paddington
Marylebone
Euston Square
Warren Street
Regent's Park
Goodge Street
Oxford Circus
Tottenham Court Road
Green Park
Piccadilly Circus
Charing Cross
Westminster
Waterloo
Embankment
Temple
Blackfriars
Mansion House
Cannon Street
Holborn
Covent Garden
Leicester Square
Russell Square
Chancery Lane
St. Paul's
Bank
Monument
Tower Hill
Aldgate
Aldgate East
Liverpool Street
Moorgate
Old Street
Angel
Farringdon
Barbican
Notting Hill Gate
Queensway
Bayswater
Lancaster Gate
Marble Arch
Bond Street
High Street Kensington
Hyde Park Corner
Knightsbridge
South Kensington
Gloucester Road
Victoria
Sloane Square
St. James's Park
Pimlico
Vauxhall
Oval
Kennington
Clapham North
Clapham Common
Clapham South
Balham
Tooting Bec
Tooting Broadway
Colliers Wood
South Wimbledon
Morden
Holland Park
Shepherd's Bush
Latimer Road
Ladbroke Grove
Westbourne Park
Royal Oak
Warwick Avenue
Maida Vale
Kilburn Park

A Bit of History

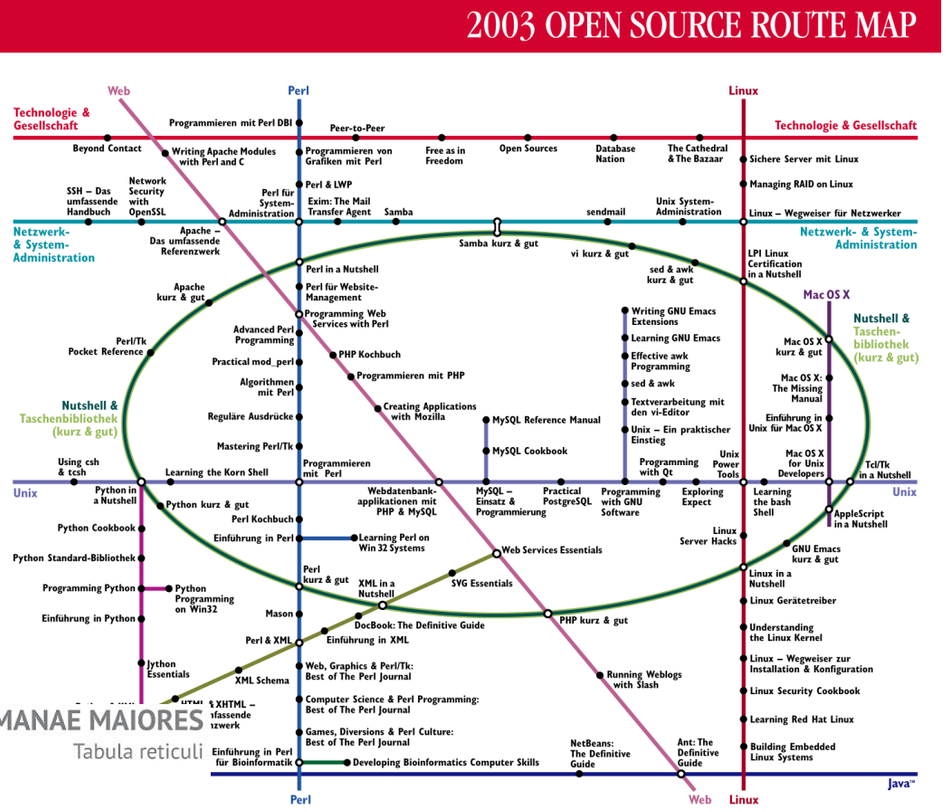
Berlin 1931
(redrawn by Maxwell Roberts)



A Bit of History









Complexity

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

Proof Sketch.

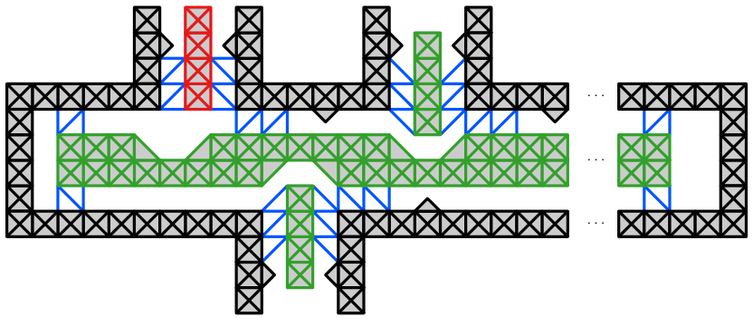
Reduction from Boolean satisfiability problem Planar-3Sat using rigid "mechanical" gadgets.

Complexity

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

Proof Sketch.

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



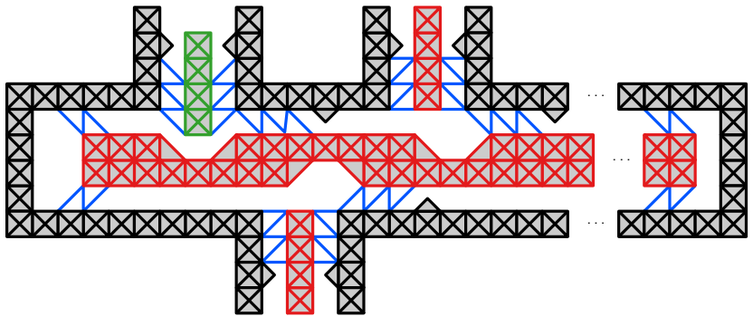
variable gadget (true)

Complexity

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

Proof Sketch.

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



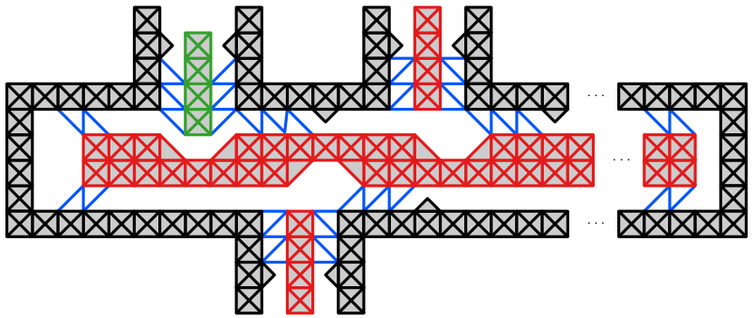
variable gadget (false)

Complexity

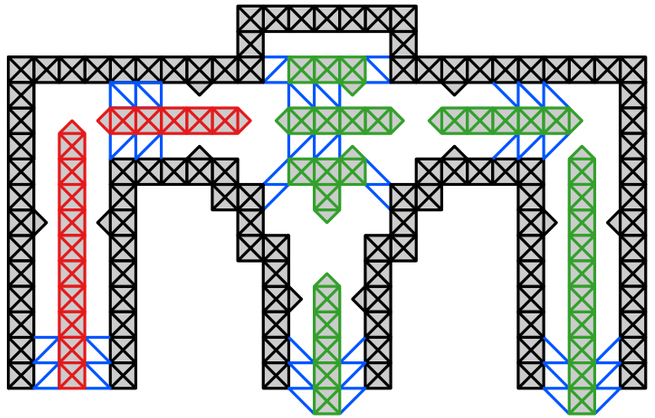
Theorem 1. [Nöllenburg 2005]
For an embedded graph with vertex degrees at most 8, **bend minimization** (R3) is NP-hard if **preserving topology** (R1) and **octilinearity** (R2) are required.

Proof Sketch.

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



variable gadget (false)



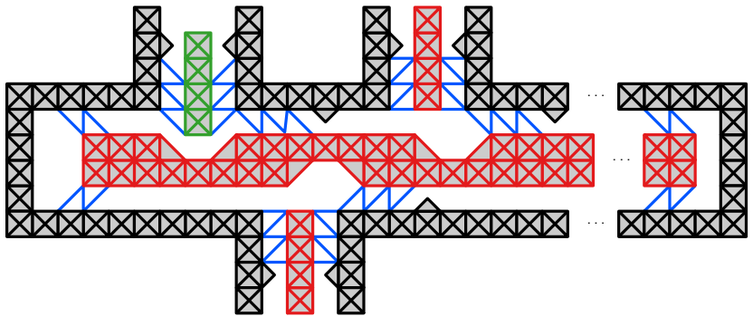
clause gadget (false, true, true)

Complexity

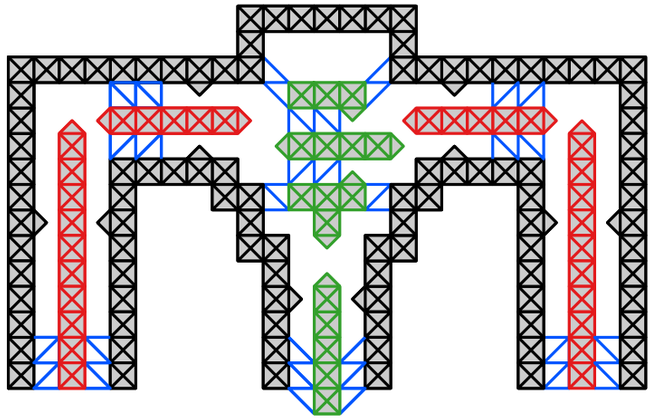
Theorem 1. [Nöllenburg 2005]
For an embedded graph with vertex degrees at most 8, **bend minimization** (R3) is NP-hard if **preserving topology** (R1) and **octilinearity** (R2) are required.

Proof Sketch.

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



variable gadget (false)



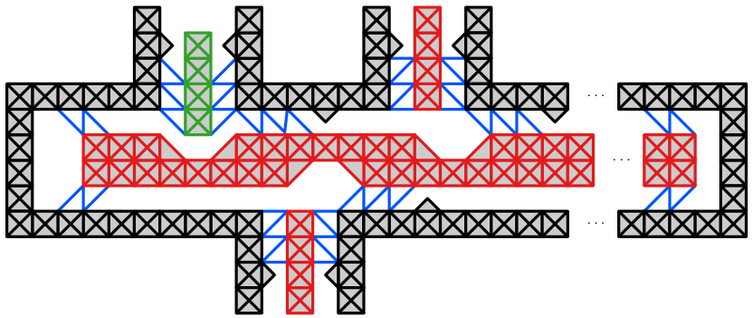
clause gadget (false, true, false)

Complexity

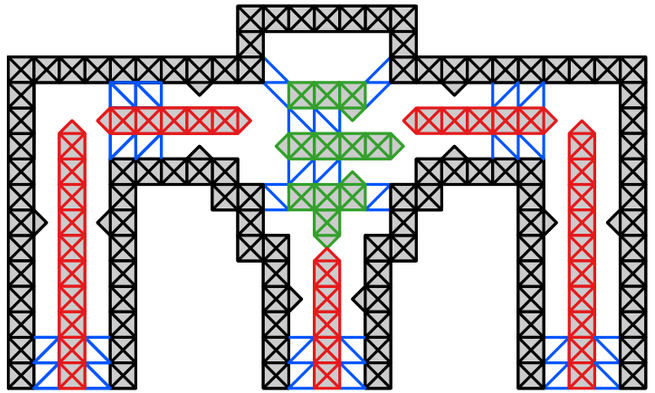
Theorem 1. [Nöllenburg 2005]
For an embedded graph with vertex degrees at most 8, **bend minimization** (R3) is NP-hard if **preserving topology** (R1) and **octilinearity** (R2) are required.

Proof Sketch.

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



variable gadget (false)



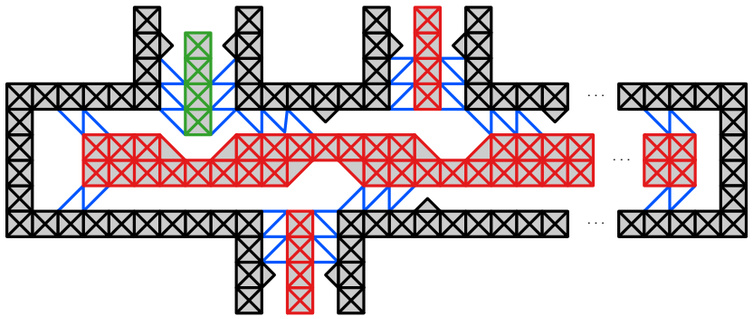
clause gadget (false, false, false)

Complexity

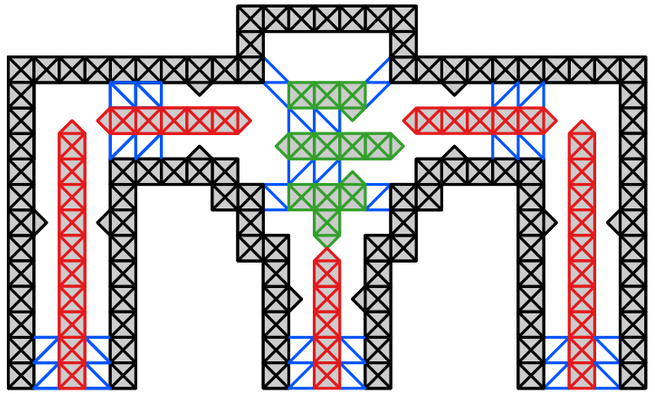
Theorem 1. [Nöllenburg 2005]
For an embedded graph with vertex degrees at most 8, **bend minimization** (R3) is NP-hard if **preserving topology** (R1) and **octilinearity** (R2) are required.

Proof Sketch.

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



variable gadget (false)



clause gadget (false, false, false)

Remark.

- no efficient exact algorithms to expect
- same problem without diagonals (rectilinear) is efficiently solvable [Tamassia '87] (→ lecture 6)

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)**

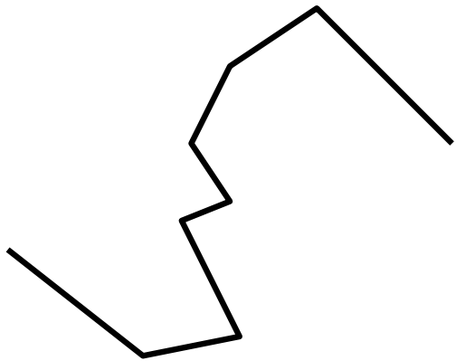
Criteria.
- minimize number of links **(R3)**

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)**

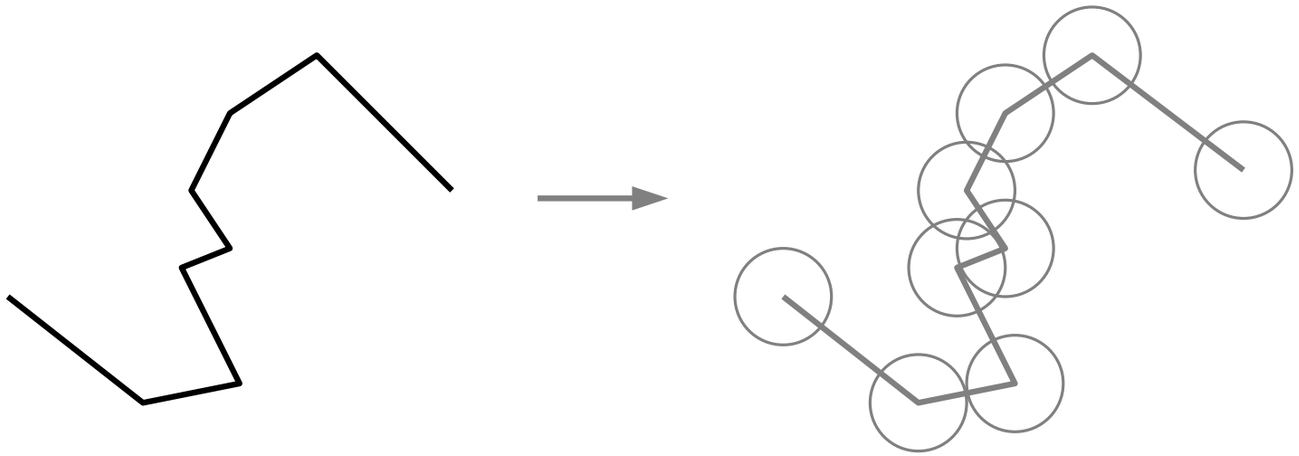


Path-Based Schematization

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)**

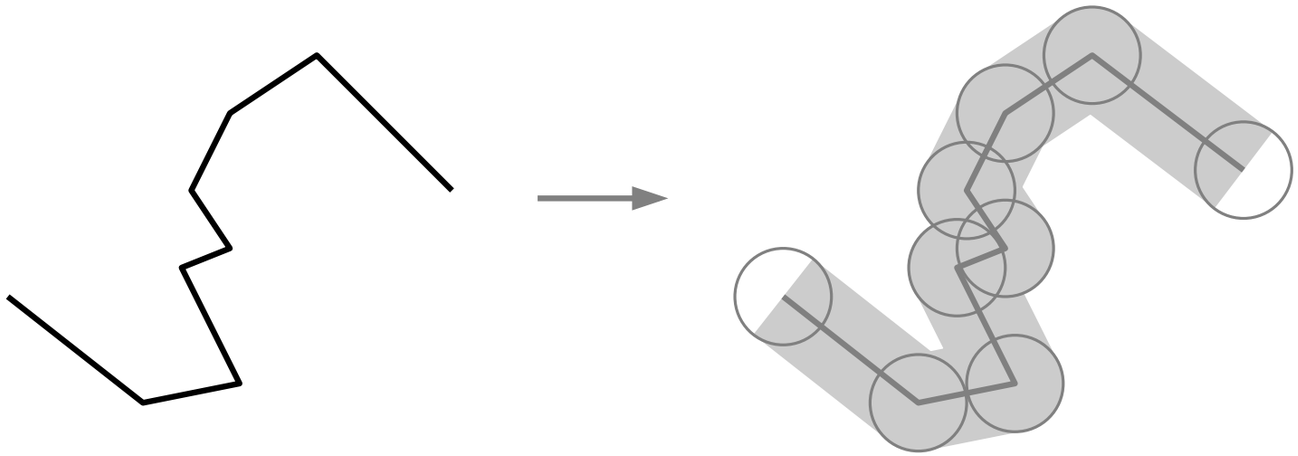


Path-Based Schematization

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)**

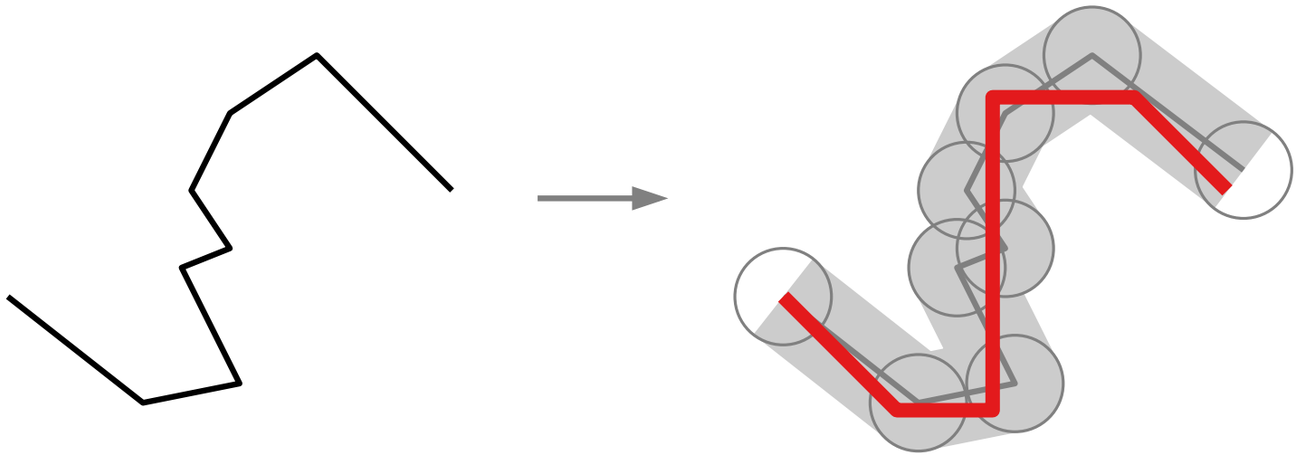


Path-Based Schematization

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

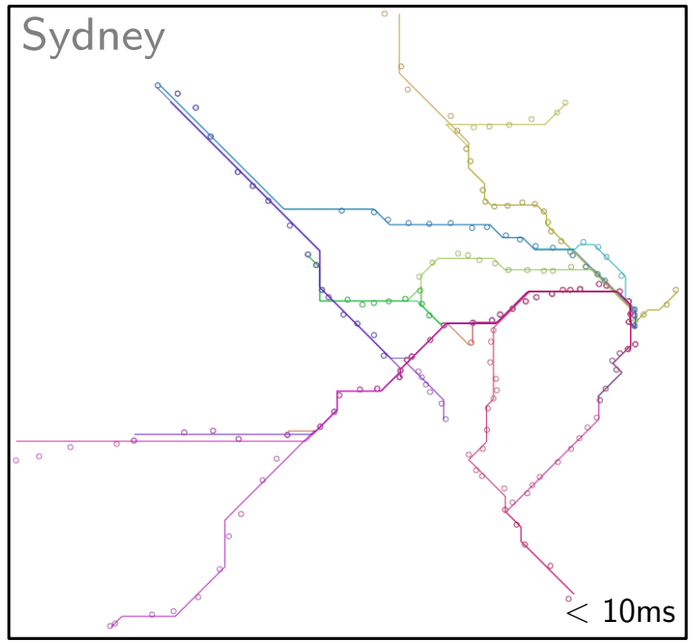
Constraints.
- $\mathcal{C}$-oriented edges (e.g. octilinear) **(R2)**
- bounded displacement **(R6)**

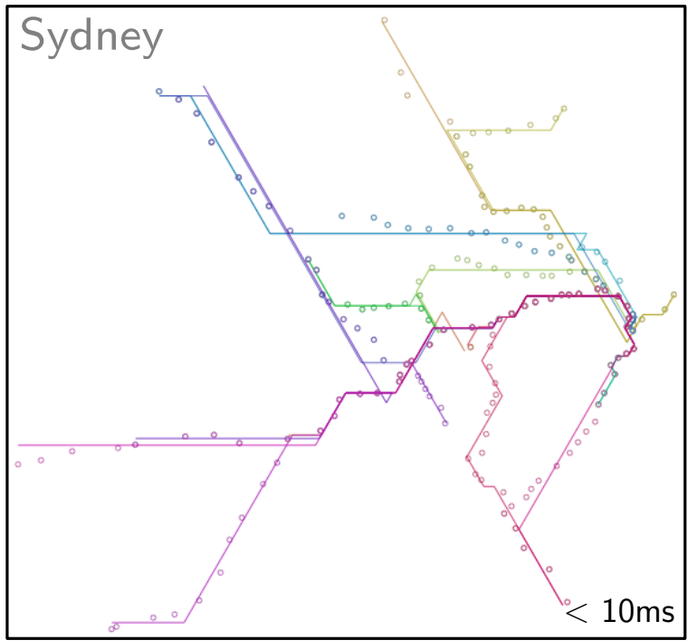
Criteria.
- minimize number of links **(R3)**



Path-based schematization – example

Theorem 2. [Dwyer, Hurst, Merrick 2008]
For a path P of length n and an 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, Gemsa, Nöllenburg, Pajor 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, Gemsa, Nöllenburg, Pajor 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 relationships of all vertices

$\mathcal{C}$-oriented Route Sketches

Theorem 3. [Delling, Gemsa, Nöllenburg, Pajor 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 relationships of all vertices

Proof.

- dynamic programming for slope assignment
- LP for length assignment

$\mathcal{C}$-oriented Route Sketches

Theorem 3. [Delling, Gemsa, Nöllenburg, Pajor 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 relationships of all vertices

Proof.

- dynamic programming for slope assignment
- LP for length assignment

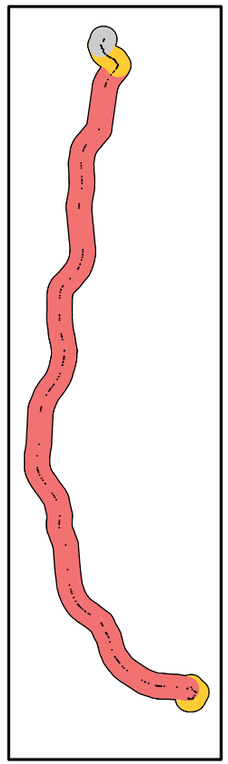
Theorem 4. [Brandes & Pampel 2009, Gemsa et al. 2011]
The d-regular (non-monotone) route sketch problem is NP-hard for any $d \geq 1$, where $\mathcal{C} = \{i \cdot 90°/d \mid i \in \mathbb{Z}\}$.

$\mathcal{C}$-oriented Route Sketches

Example. Bremen to Cuxhaven [Gemsa, Nöllenburg, Pajor, Rutter 2011]

$\mathcal{C}$-oriented Route Sketches

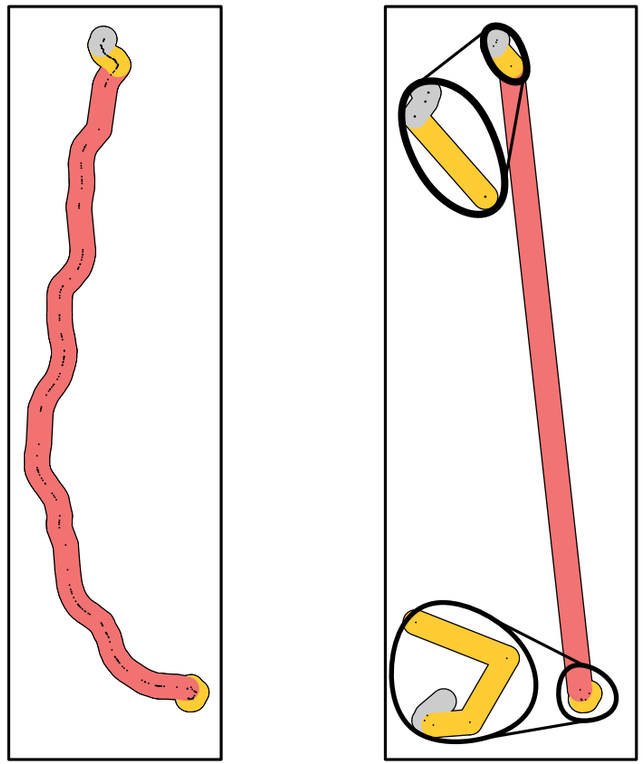
Example. Bremen to Cuxhaven [Gemsa, Nöllenburg, Pajor, Rutter 2011]



input

$\mathcal{C}$-oriented Route Sketches

Example. Bremen to Cuxhaven [Gemsa, Nöllenburg, Pajor, Rutter 2011]

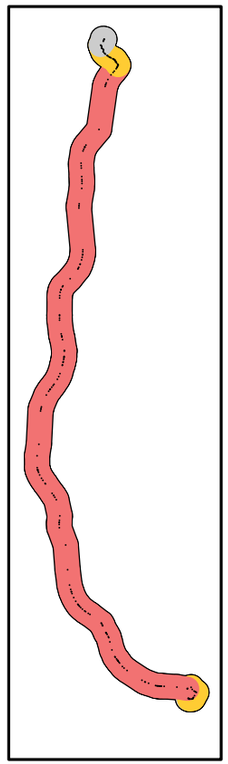


input

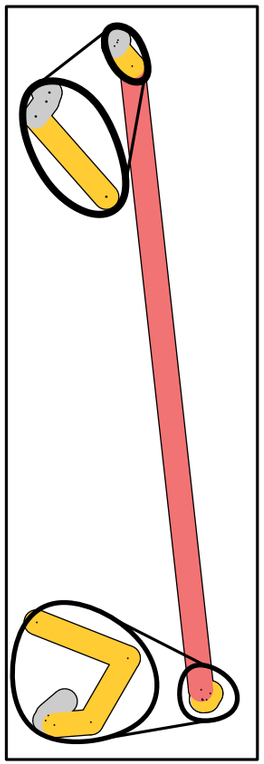
simplified

$\mathcal{C}$-oriented Route Sketches

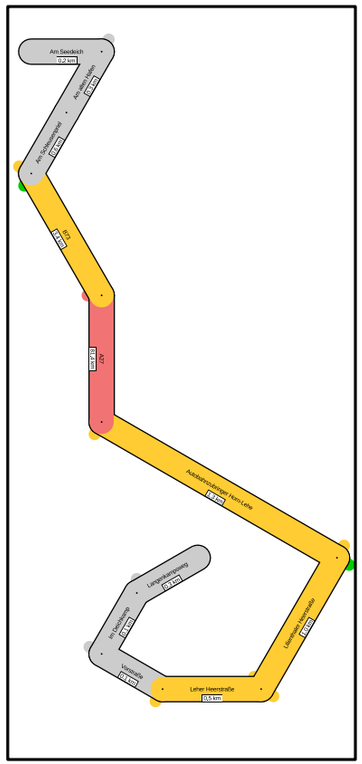
Example. Bremen to Cuxhaven [Gemsa, Nöllenburg, Pajor, Rutter 2011]



input



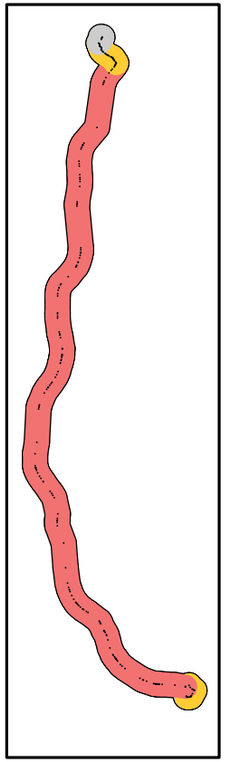
simplified



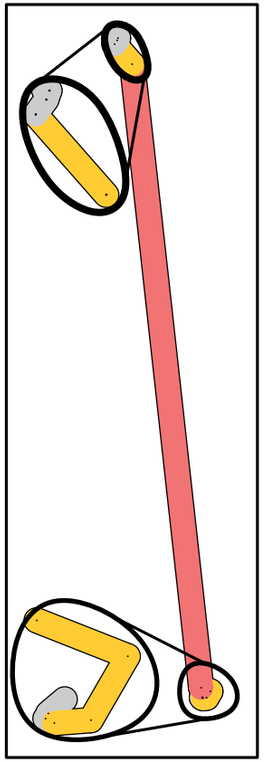
optimized by MIP

$\mathcal{C}$-oriented Route Sketches

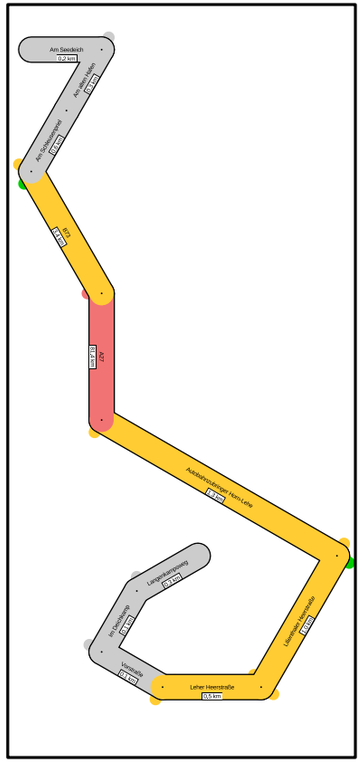
Example. Bremen to Cuxhaven [Gemsa, Nöllenburg, Pajor, Rutter 2011]



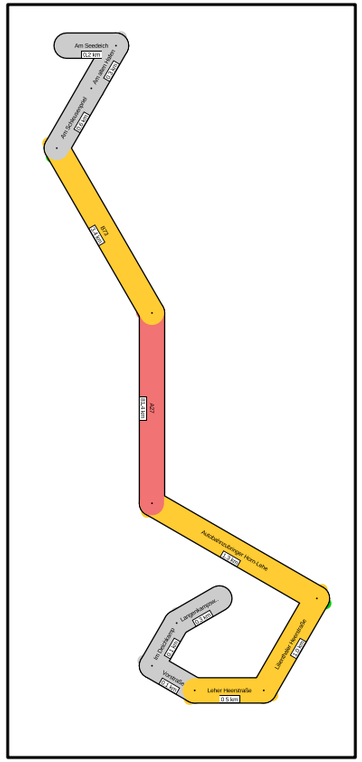
input



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.

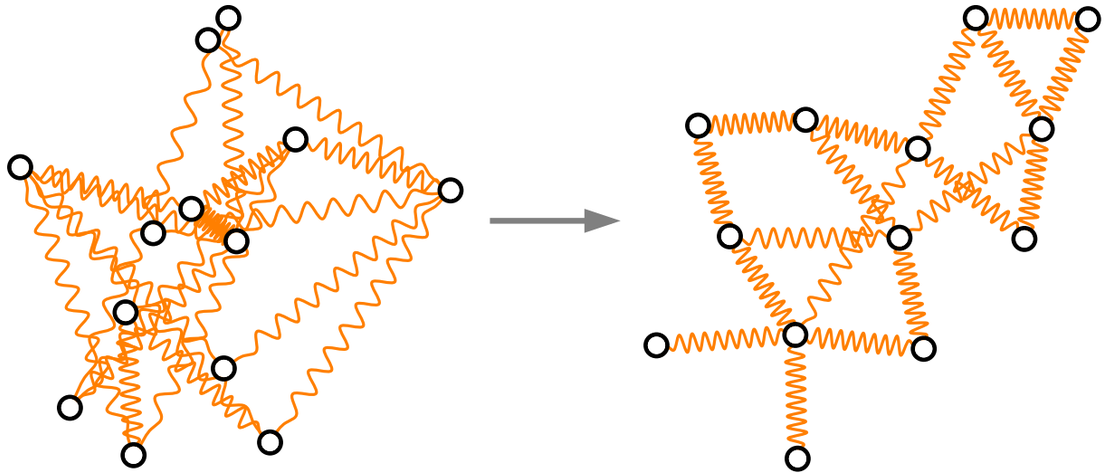
- polynomial running times
- $\mathcal{C}$-orientation and bounded displacement guaranteed
- bend minimization
- extends to metro networks:
  - decompose metro network into paths
  - schematize individual paths
  - glue schematized paths together at interchanges

Cons.

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

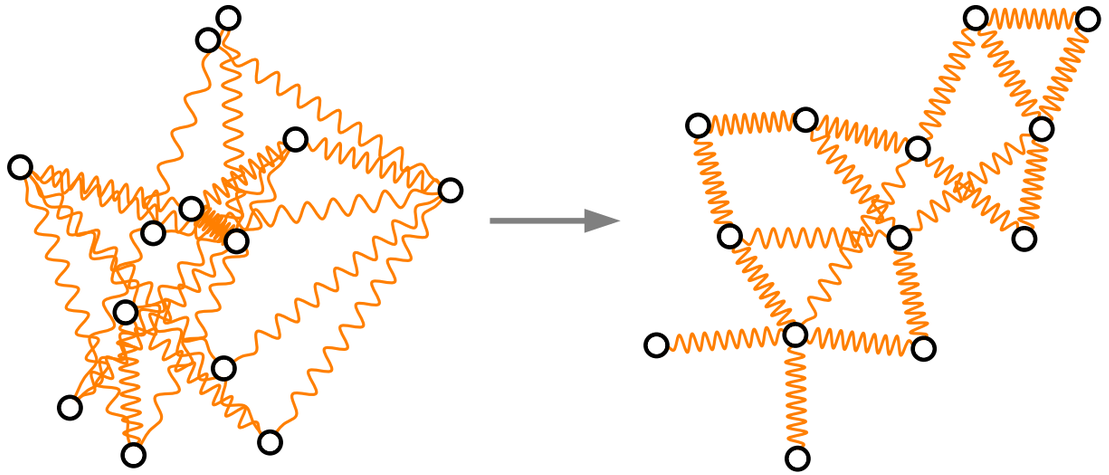
Force-Based Schematization

Idea. Apply well-known force-based graph drawing approach ($\rightarrow$ lecture 2)



Force-Based Schematization

Idea. Apply well-known force-based graph drawing approach ($\rightarrow$ lecture 2)

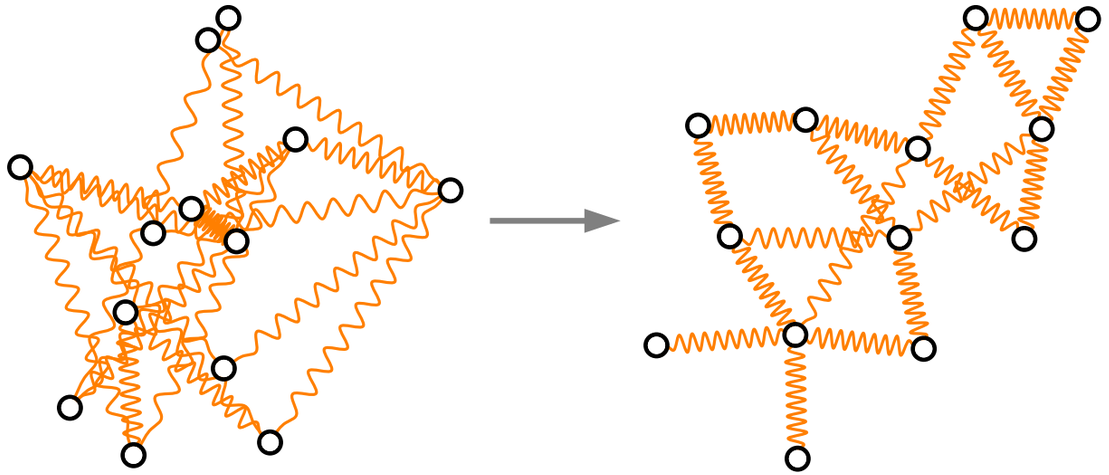


Recall.

- vertices are charged particles repelling each other

Force-Based Schematization

Idea. Apply well-known force-based graph drawing approach ($\rightarrow$ lecture 2)

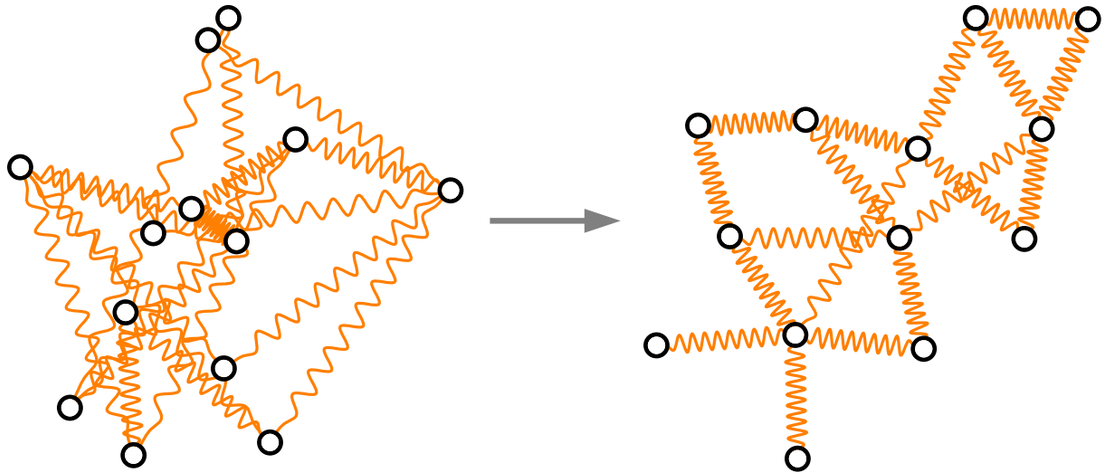


Recall.

- 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 ($\rightarrow$ lecture 2)

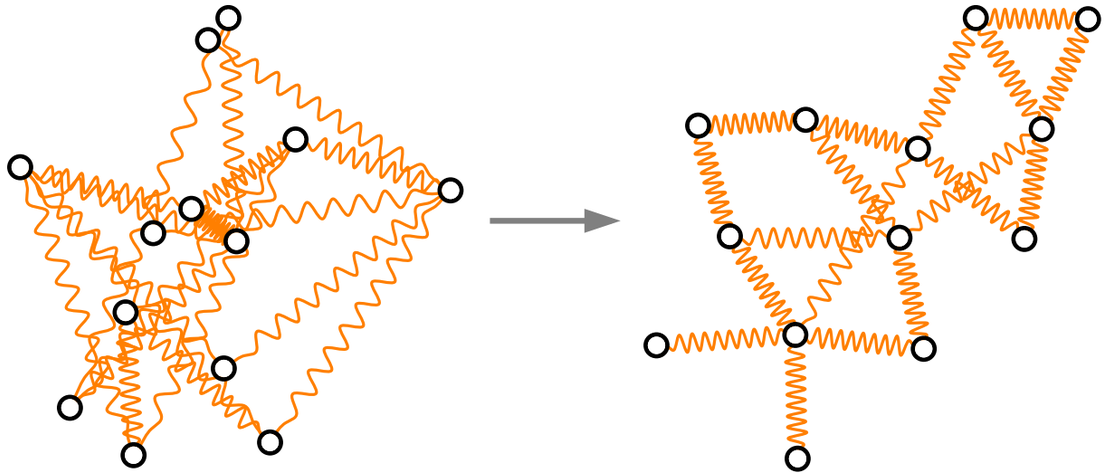


Recall.

- 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 ($\rightarrow$ lecture 2)



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, Merrick, do Nascimento 2006]

- contract degree-2 vertices into weighted edges

Force-Based Schematization – Octilinear

[Hong, Merrick, do Nascimento 2006]

- contract degree-2 vertices into weighted edges **(R3)** (Draw metro lines simple and monotone.)

Force-Based Schematization – Octilinear

[Hong, Merrick, do Nascimento 2006]

- contract degree-2 vertices into weighted edges **(R3)** (Draw metro lines simple and monotone.)
- define octilinear magnetic field attracting edges **(R2)** (Restrict edge orientations.)

Force-Based Schematization – Octilinear

[Hong, Merrick, do Nascimento 2006]

- contract degree-2 vertices into weighted edges **(R3)** (Draw metro lines simple and monotone.)
- define octilinear magnetic field attracting edges **(R2)** (Restrict edge orientations.)
- only apply topology preserving vertex moves **(R1)** (Do not change the network topology.)

Force-Based Schematization – Octilinear

[Hong, Merrick, do Nascimento 2006]

- contract degree-2 vertices into weighted edges **(R3)** (Draw metro lines simple and monotone.)
- define octilinear magnetic field attracting edges **(R2)** (Restrict edge orientations.)
- only apply topology preserving vertex moves **(R1)** (Do not change the network topology.)
- spring lengths model uniform edge lengths **(R7)** (Use uniform edge lengths.)

Force-Based Schematization – Octilinear

[Hong, Merrick, do Nascimento 2006]

- contract degree-2 vertices into weighted edges **(R3)** (Draw metro lines simple and monotone.)
- define octilinear magnetic field attracting edges **(R2)** (Restrict edge orientations.)
- only apply topology preserving vertex moves **(R1)** (Do not change the network topology.)
- spring lengths model uniform edge lengths **(R7)** (Use uniform edge lengths.)
- vertex repulsion models feature separation **(R8)** (Keep unrelated features apart.)

Force-Based Schematization – Octilinear

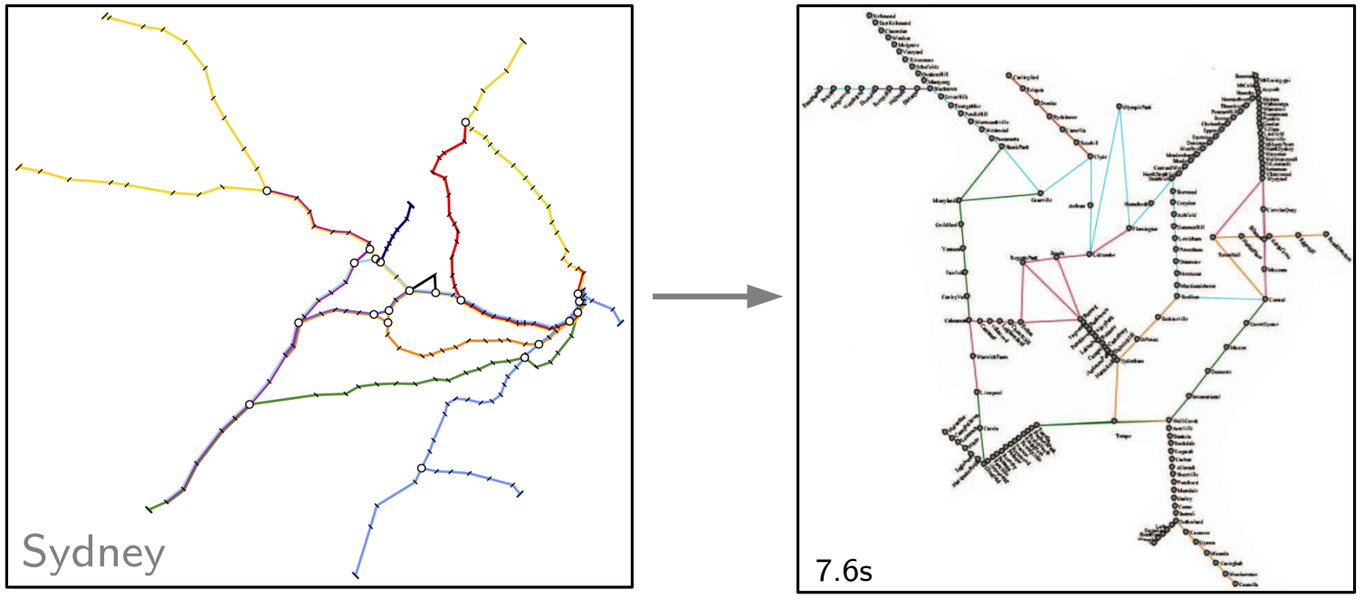
[Hong, Merrick, do Nascimento 2006]

- contract degree-2 vertices into weighted edges **(R3)** (Draw metro lines simple and monotone.)
- define octilinear magnetic field attracting edges **(R2)** (Restrict edge orientations.)
- only apply topology preserving vertex moves **(R1)** (Do not change the network topology.)
- spring lengths model uniform edge lengths **(R7)** (Use uniform edge lengths.)
- vertex repulsion models feature separation **(R8)** (Keep unrelated features apart.)
- station labels are placed in an independent second step

Force-Based Schematization – Octilinear

[Hong, Merrick, do Nascimento 2006]

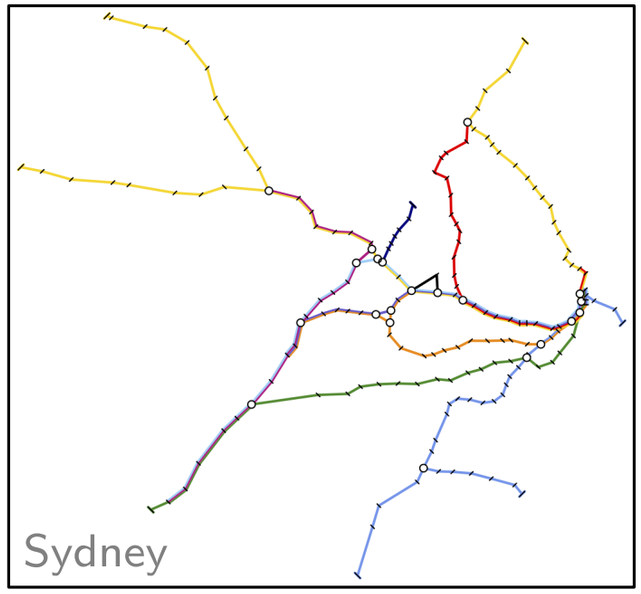
- contract degree-2 vertices into weighted edges **(R3)** (Draw metro lines simple and monotone.)
- define octilinear magnetic field attracting edges **(R2)** (Restrict edge orientations.)
- only apply topology preserving vertex moves **(R1)** (Do not change the network topology.)
- spring lengths model uniform edge lengths **(R7)** (Use uniform edge lengths.)
- vertex repulsion models feature separation **(R8)** (Keep unrelated features apart.)
- station labels are placed in an independent second step

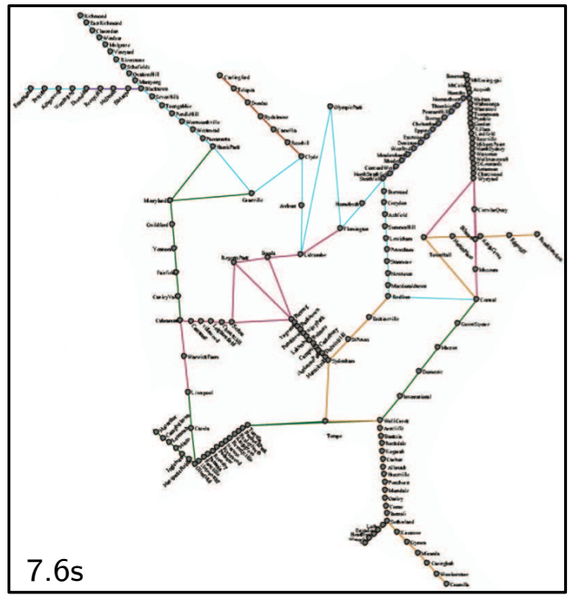


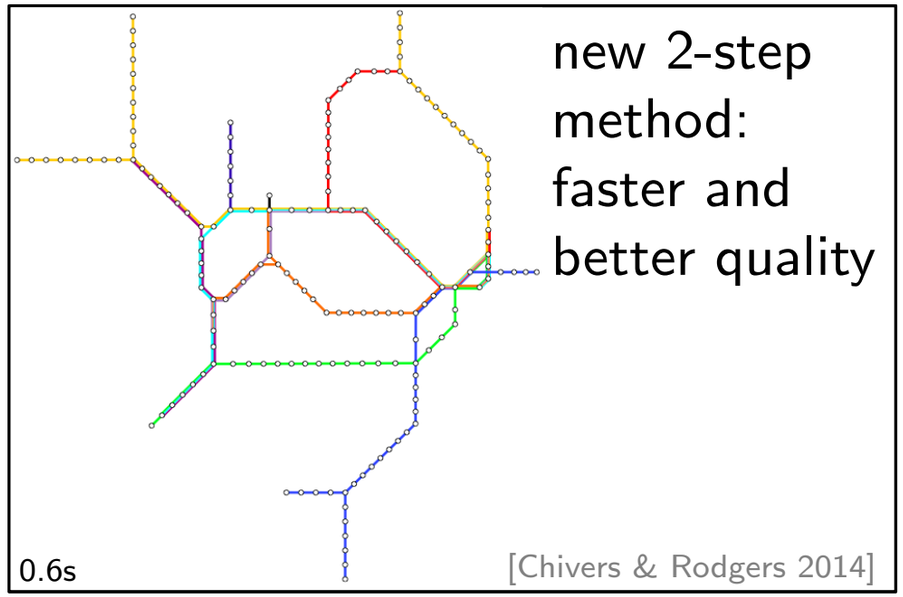
Force-Based Schematization – Octilinear

[Hong, Merrick, do Nascimento 2006]

- contract degree-2 vertices into weighted edges **(R3)** (Draw metro lines simple and monotone.)
- define octilinear magnetic field attracting edges **(R2)** (Restrict edge orientations.)
- only apply topology preserving vertex moves **(R1)** (Do not change the network topology.)
- spring lengths model uniform edge lengths **(R7)** (Use uniform edge lengths.)
- vertex repulsion models feature separation **(R8)** (Keep unrelated features apart.)
- station labels are placed in an independent second step







Force-Based Schematization – Bézier

[Fink, Haverkort, Nöllenburg, Roberts, Schuhmann, Wolff 2012]

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

Force-Based Schematization – Bézier

[Fink, Haverkort, Nöllenburg, Roberts, Schuhmann, Wolff 2012]

- 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) (Let lines pass straight through interchanges.)

Force-Based Schematization – Bézier

[Fink, Haverkort, Nöllenburg, Roberts, Schuhmann, Wolff 2012]

- 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) (Let lines pass straight through interchanges.)
- (standard) attractive and repulsive forces

  (R7)+(R8) (Uniform edge lengths. + Keep unrelated features apart.)

Force-Based Schematization – Bézier

[Fink, Haverkort, Nöllenburg, Roberts, Schuhmann, Wolff 2012]

- 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) (Let lines pass straight through interchanges.)
- (standard) attractive and repulsive forces
  (R7)+(R8) (Uniform edge lengths. + Keep unrelated features apart.)
- (weak) force towards initial position **(R6)** (Minimize geometric distortion & displacement.)

Force-Based Schematization – Bézier

[Fink, Haverkort, Nöllenburg, Roberts, Schuhmann, Wolff 2012]

- 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) (Let lines pass straight through interchanges.)
- (standard) attractive and repulsive forces
  (R7)+(R8) (Uniform edge lengths. + Keep unrelated features apart.)
- (weak) force towards initial position **(R6)** (Minimize geometric distortion & displacement.)
- forces improving curve shape (low curvature)
  (R3) (Draw metro lines as simple and monotone as possible.)

Force-Based Schematization – Bézier

[Fink, Haverkort, Nöllenburg, Roberts, Schuhmann, Wolff 2012]

- 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) (Let lines pass straight through interchanges.)
- (standard) attractive and repulsive forces
  (R7)+(R8) (Uniform edge lengths. + Keep unrelated features apart.)
- (weak) force towards initial position **(R6)** (Minimize geometric distortion & displacement.)
- forces improving curve shape (low curvature)
  (R3) (Draw metro lines as simple and monotone as possible.)
- merge curves whenever possible **(R3)**

Force-Based Schematization – Bézier

[Fink, Haverkort, Nöllenburg, Roberts, Schuhmann, Wolff 2012]

- 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) (Let lines pass straight through interchanges.)
- (standard) attractive and repulsive forces
  (R7)+(R8) (Uniform edge lengths. + Keep unrelated features apart.)
- (weak) force towards initial position **(R6)** (Minimize geometric distortion & displacement.)
- forces improving curve shape (low curvature)
  (R3) (Draw metro lines as simple and monotone as possible.)
- merge curves whenever possible **(R3)**
- forces improving angular resolution **(R5)** (Use large angular resolution in stations.)

Force-Based Schematization – Bézier

[Fink, Haverkort, Nöllenburg, Roberts, Schuhmann, Wolff 2012]

- 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) (Let lines pass straight through interchanges.)
- (standard) attractive and repulsive forces
  (R7)+(R8) (Uniform edge lengths. + Keep unrelated features apart.)
- (weak) force towards initial position **(R6)** (Minimize geometric distortion & displacement.)
- forces improving curve shape (low curvature)
  (R3) (Draw metro lines as simple and monotone as possible.)
- merge curves whenever possible **(R3)**
- forces improving angular resolution **(R5)** (Use large angular resolution in stations.)
- apply topology-preserving moves only **(R1)** (Do not change the network topology.)

Force-Based Schematization – Bézier

[Fink, Haverkort, Nöllenburg, Roberts, Schuhmann, Wolff 2012]

- 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) (Let lines pass straight through interchanges.)

- (standard) attractive and repulsive forces

(R7) + **(R8)** (Uniform edge lengths. + Keep unrelated features apart.)

- (weak)

(R6) (Minimize geometric distortion & displacement.)

- forces i

urvature)

raw metro lines as simple and monotone as possible.)

- merge

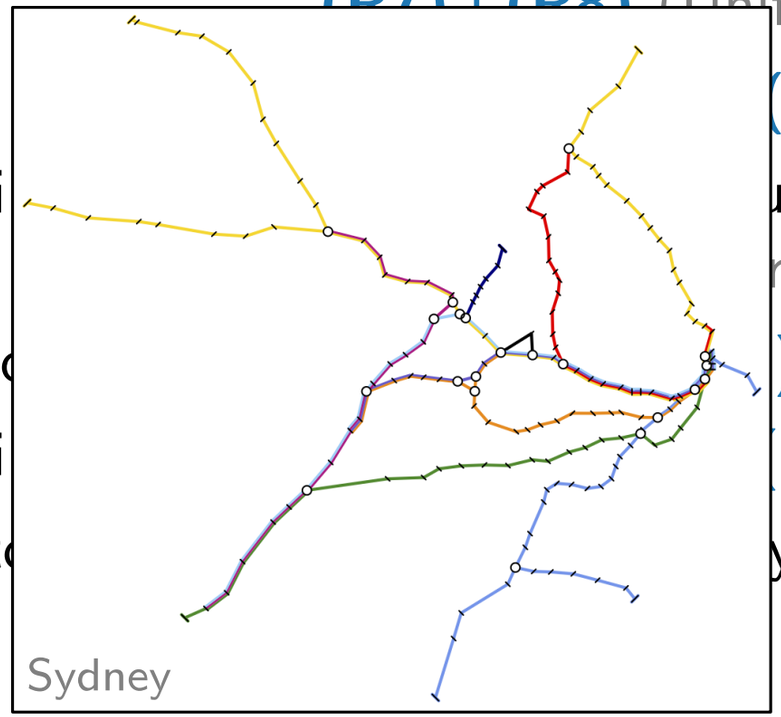
)

- forces i

(R5) (Use large angular resolution in stations.)

- apply t

y **(R1)** (Do not change the network topology.)



Force-Based Schematization – Bézier

[Fink, Haverkort, Nöllenburg, Roberts, Schuhmann, Wolff 2012]

- convert (octilinear) input layout into Bézier curves
  $\rightarrow$ vertices and control points, both subject to forces
- metro line subcurves share tangents in common vertex
- (standard) attractive and repulsive forces
- (weak)
- forces i
- merge
- forces i
- apply t

(R4) (Let lines pass straight through interchanges.)

(R7) + **(R8)** (Uniform edge lengths + Keep unrelated features apart.)

(R6) (M & displacement.)

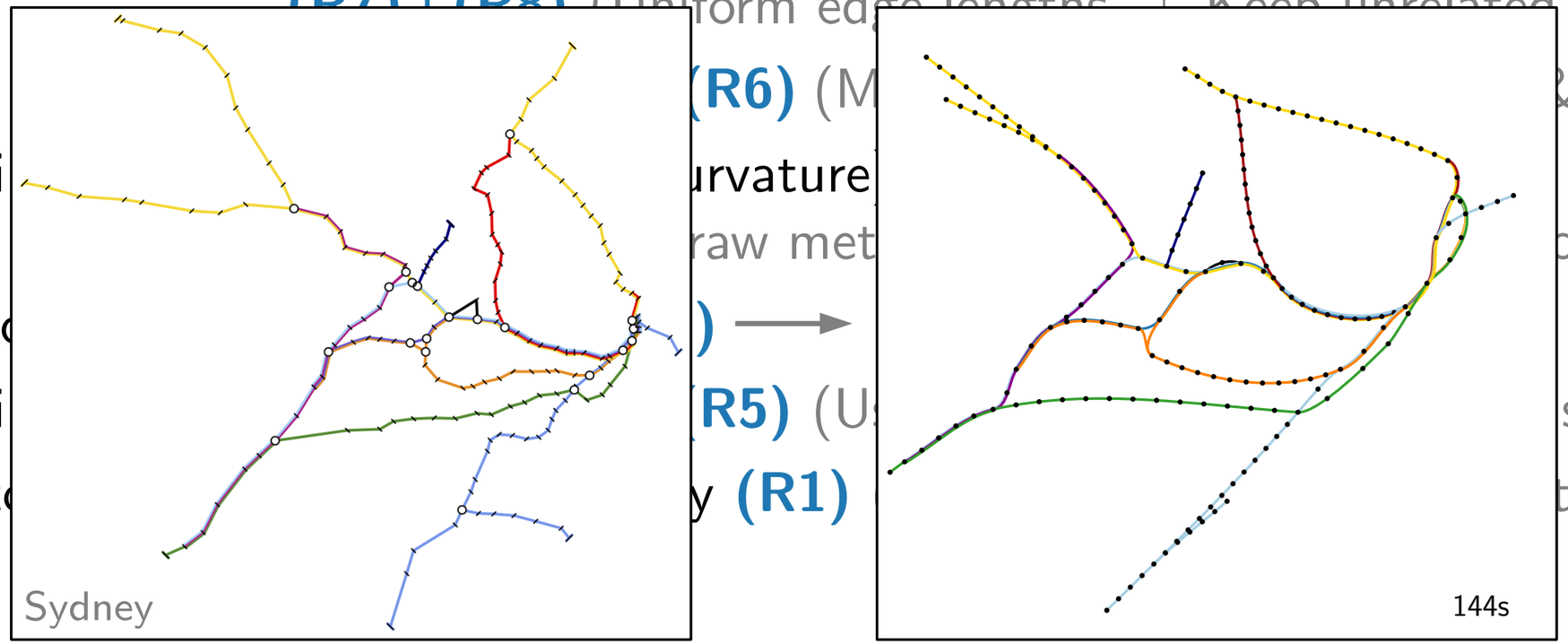
urvature

raw met one as possible.)

)

(R5) (U stations.)

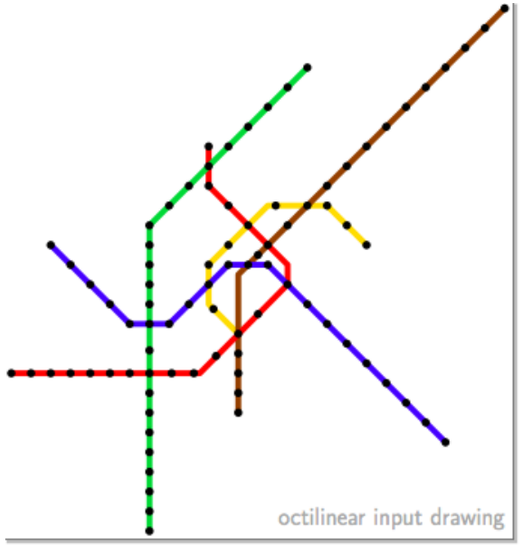
y **(R1)** topology.)

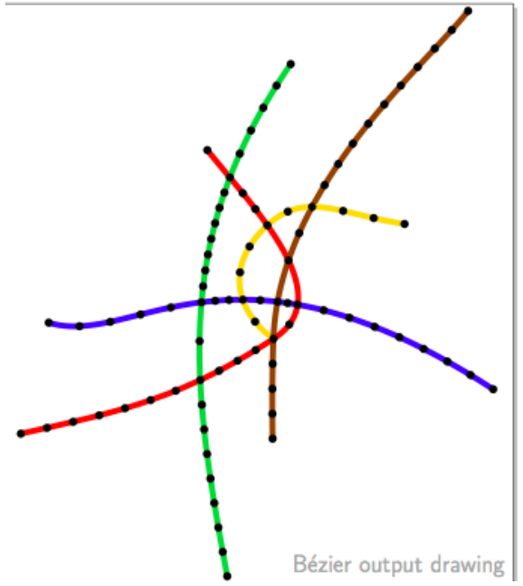


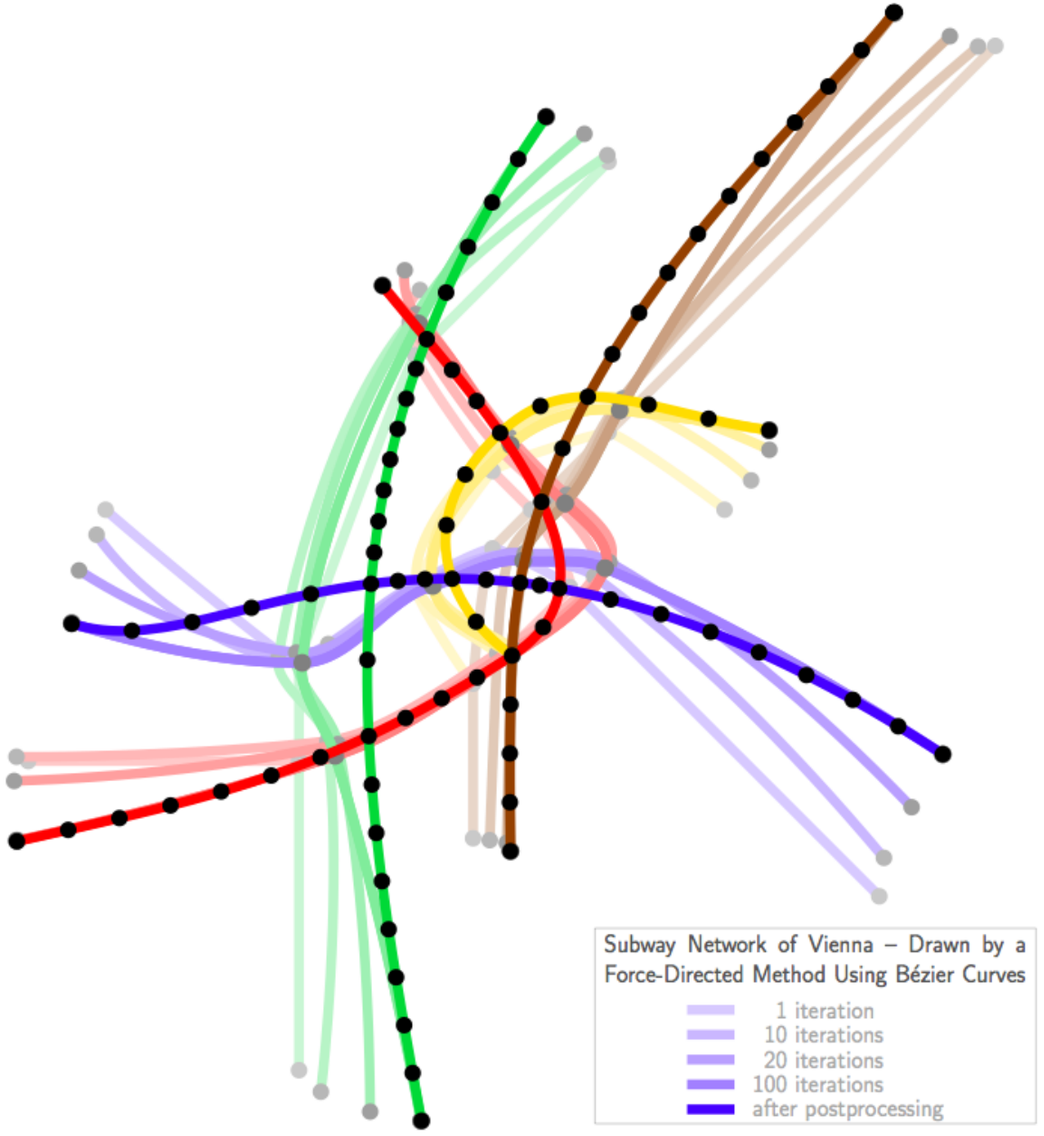
Force-Based Schematization – Bézier

[Fink, Haverkort, Nöllenburg, Roberts, Schuhmann, Wolff 2012]

Vienna:

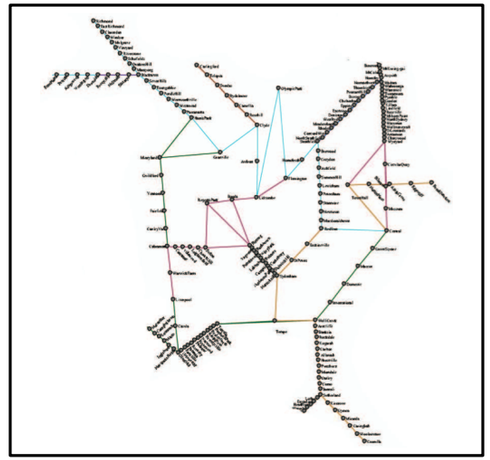






Force-Based Schematization – Discussion

Octilinear.



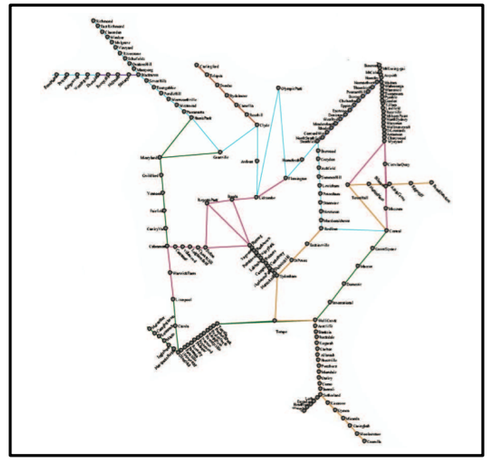
Bézier.



Force-Based Schematization – Discussion

Octilinear.

- guarantees topology **(R1)**
- slower than path-based algorithms, but still fast enough
- no strict enforcement of octilinearity **(R2)**
- quite unbalanced edge lengths **(R7)**
- bends in interchanges **(R4)**
- no distortion restriction **(R6)**

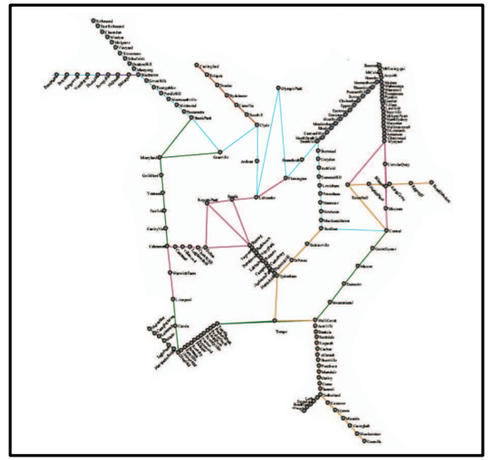


Bézier.



Force-Based Schematization – Discussion

Octilinear.



- guarantees topology **(R1)**
- slower than path-based algorithms, but still fast enough
- no strict enforcement of octilinearity **(R2)**
- quite unbalanced edge lengths **(R7)**
- bends in interchanges **(R4)**
- no distortion restriction **(R6)**

Bézier.



- guarantees topology **(R1)**
- first curvilinear metro map algorithm
- takes almost all design rules into account
- sometimes small angles and distances **(R5)+(R8)**
- works well on small and medium instances, but difficulties with more complex networks

Local Schematization

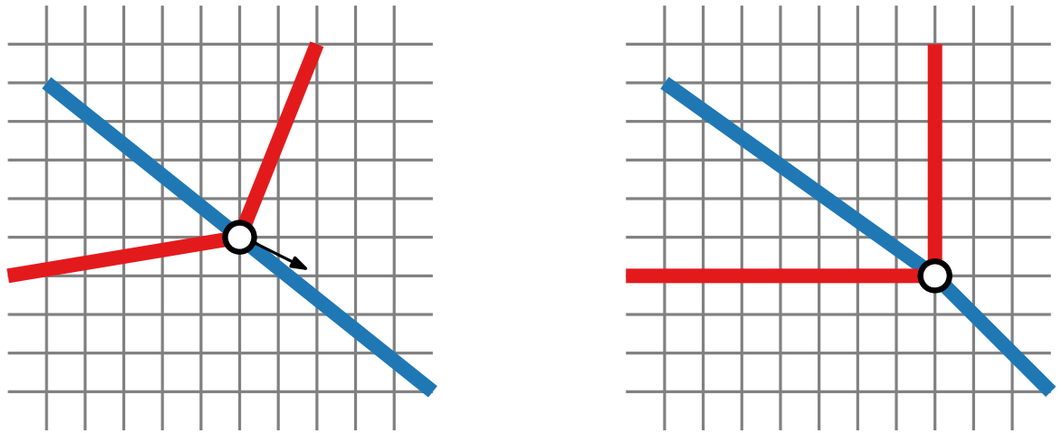
Idea.

- define a (multi-criteria) layout quality function

Local Schematization

Idea.

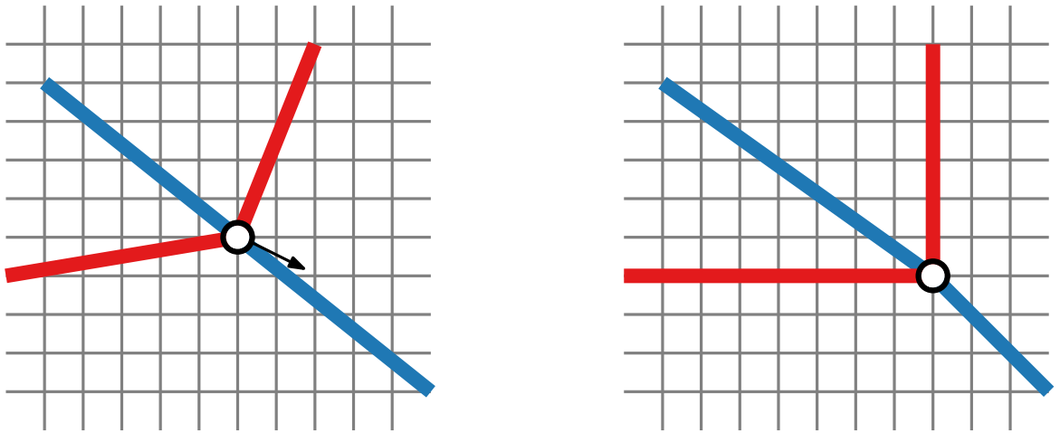
- define a (multi-criteria) layout quality function
- improve layout quality step-by-step by locally moving vertices to better nearby (grid) positions



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)**

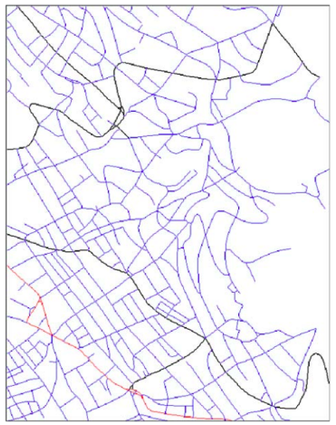
Local Schematization

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)**



Zürich

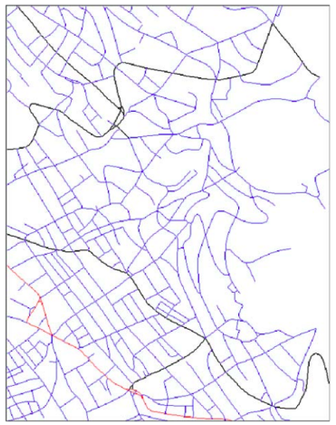
Local Schematization

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)**



Zürich



100

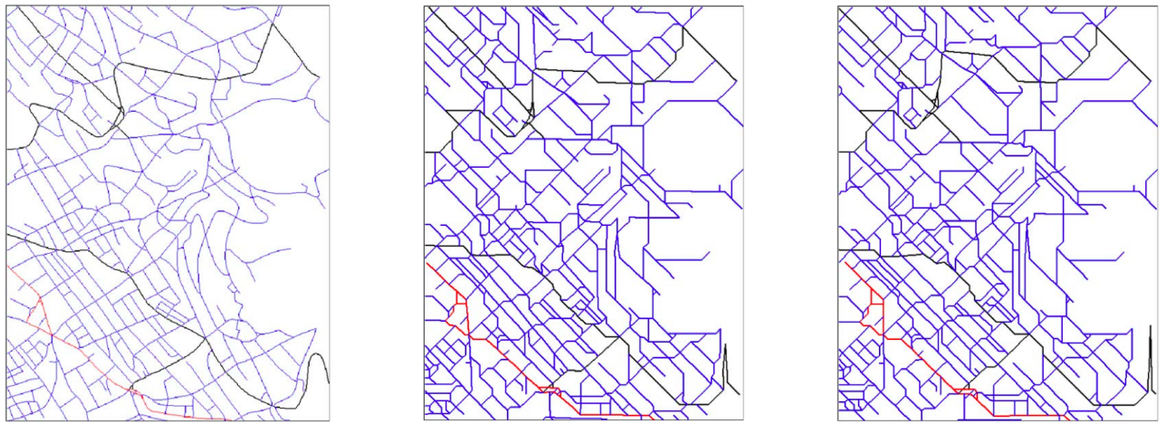
Local Schematization

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)**



Zürich 100 1,000

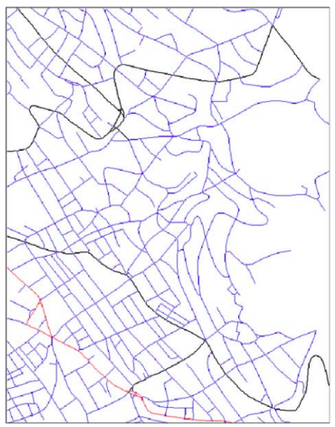
Local Schematization

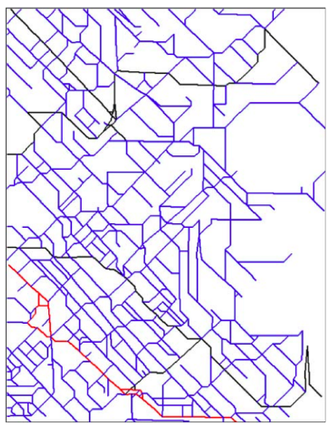
Idea.

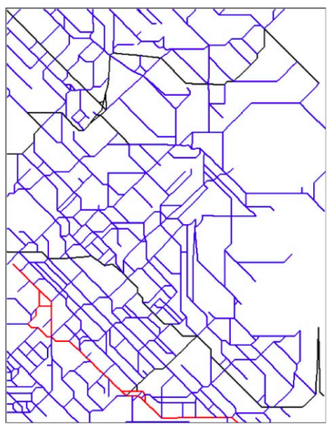
- define a (multi-criteria) layout quality function
- improve layout quality step-by-step by locally moving vertices to better nearby (grid) positions
- applicable optimization techniques: hill climbing, simulated annealing, ant colony optimization, ...

[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)**


Zürich


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, Taylor, Anand, Thomas 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, Taylor, Anand, Thomas 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, Taylor, Anand, Thomas 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, ...

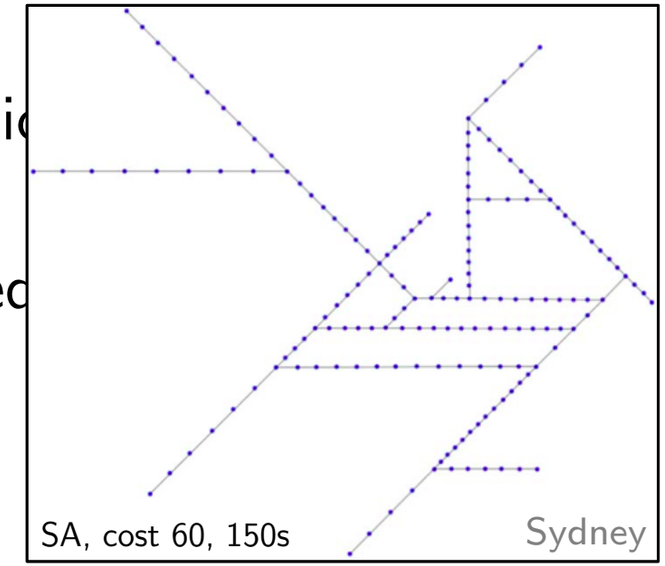
[Ware, Taylor, Anand, Thomas 2006, Ware & Richards 2013]

- 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 vertic to better nearby (grid) positions
- applicable optimization techniques: hill climbing, simulated annealing, ant colony optimization, ...



[Ware, Taylor, Anand, Thomas 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)

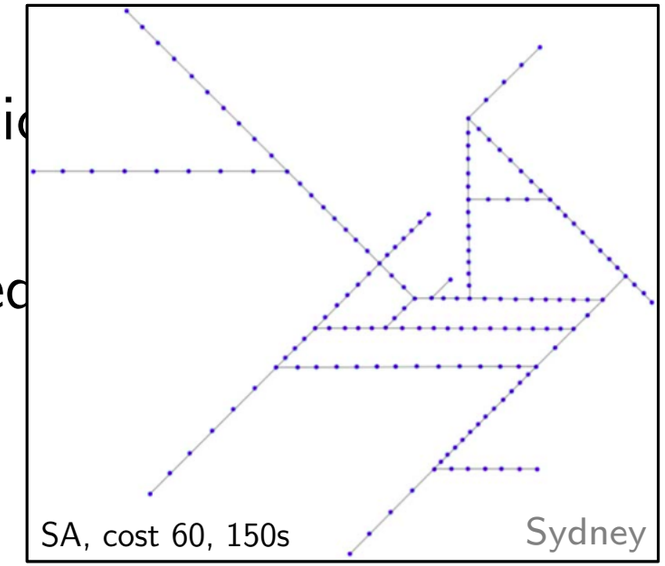
Local Schematization

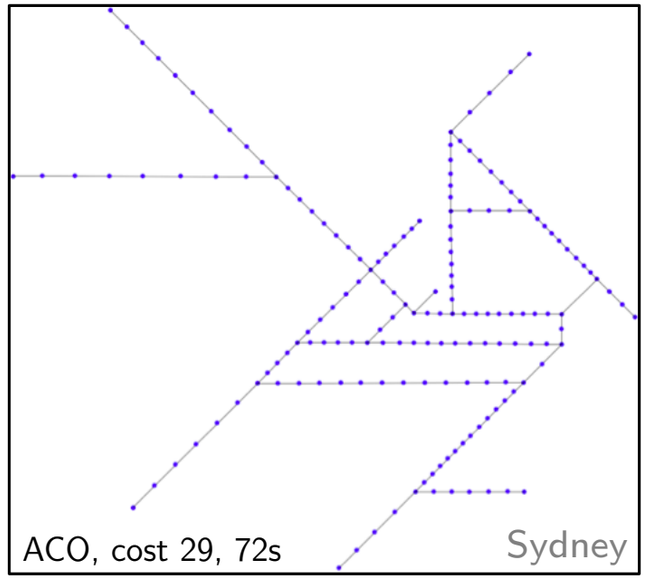
Idea.

- 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, Taylor, Anand, Thomas 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, Rodgers, Burkhard, Meier, Smis 2011]

Idea.

- design rules as before

Local Schematization

[Stott, Rodgers, Burkhard, Meier, Smis 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, Rodgers, Burkhard, Meier, Smis 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, Rodgers, Burkhard, Meier, Smis 2011]

Idea.

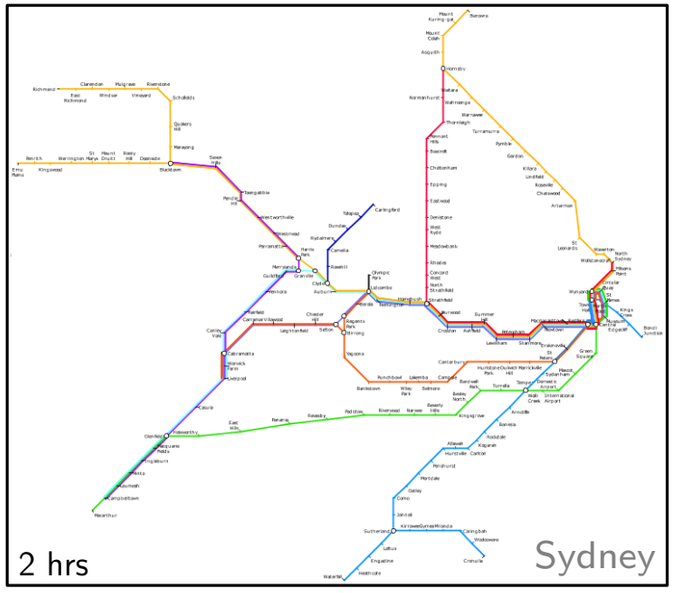
- design rules as before
- additionally include metro map specific criteria (bend minimization **(R3)**, interchange straightness **(R4)**, angular resolution **(R5)**, relative positions **(R6)**)
- integrate alternating label placement rounds
- some ad-hoc fixes for local minima situations

Local Schematization

[Stott, Rodgers, Burkhard, Meier, Smis 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

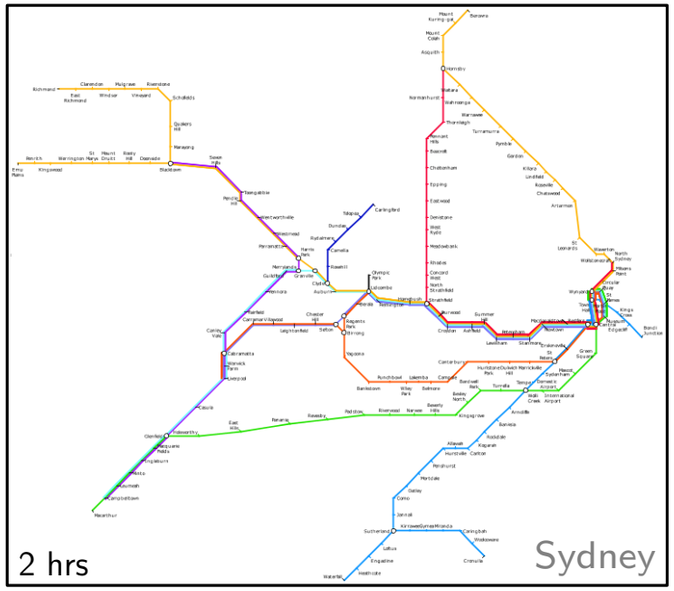


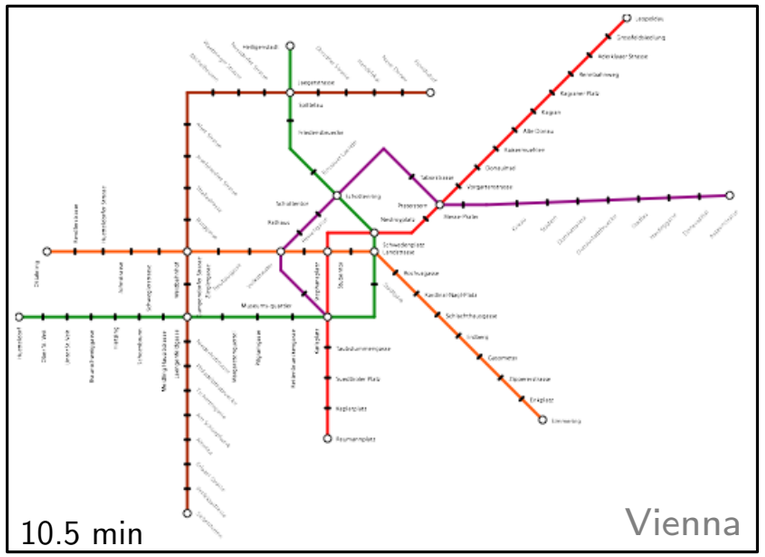
Local Schematization

[Stott, Rodgers, Burkhard, Meier, Smis 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

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

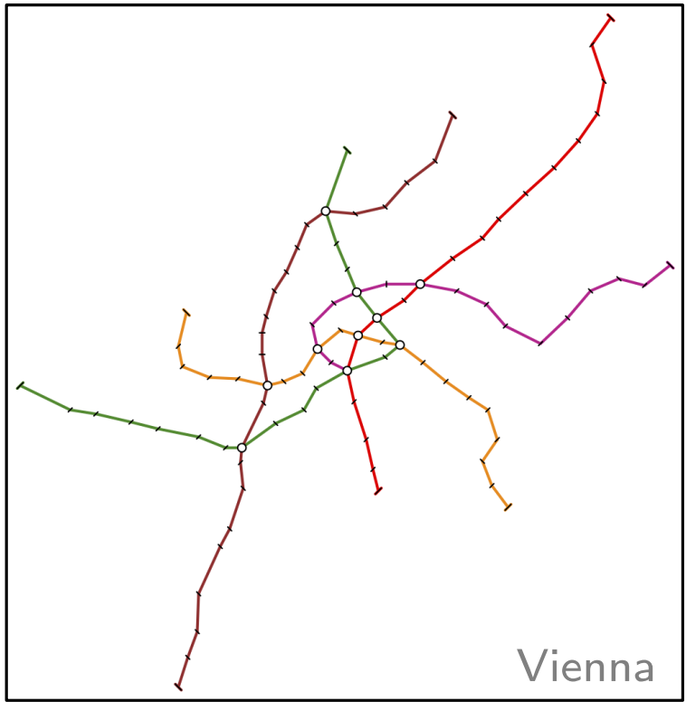
Mixed-Integer Programming

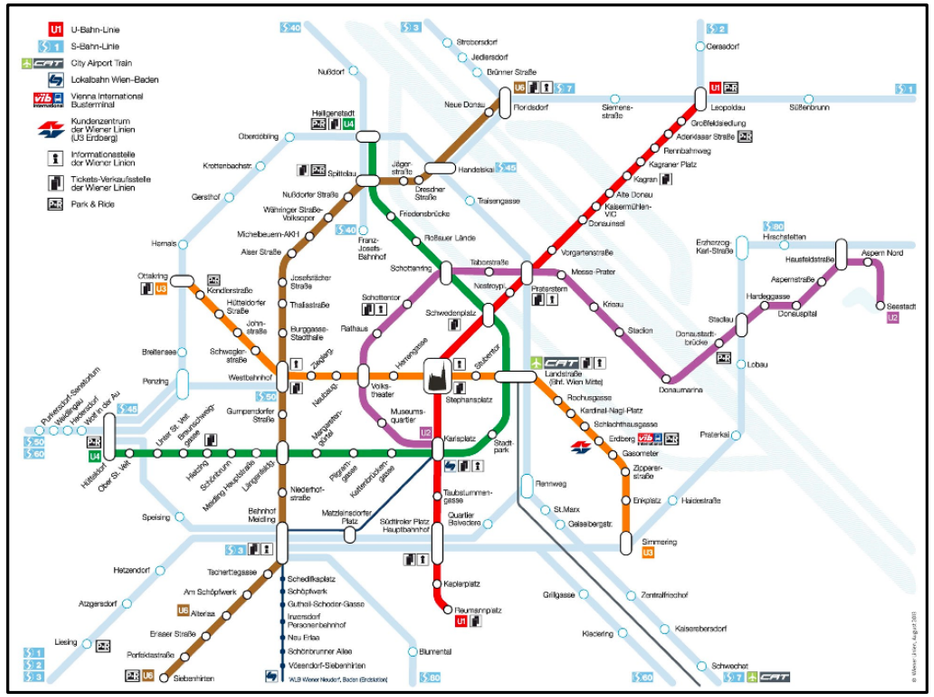
[Nöllenburg & Wolff 2011]

- 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**

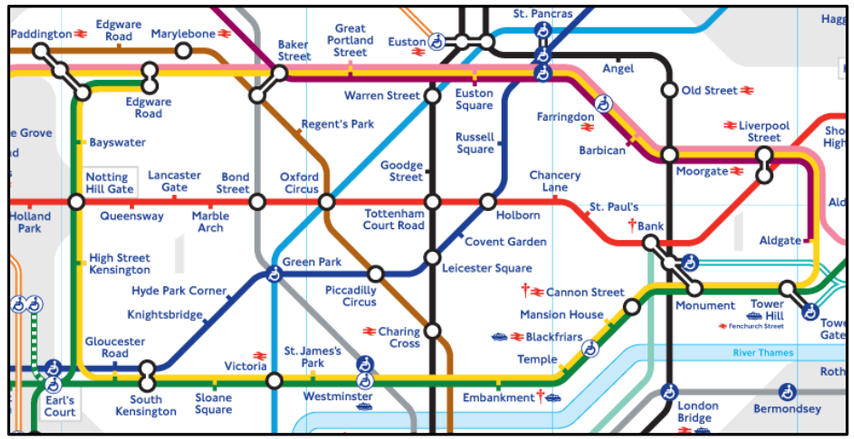


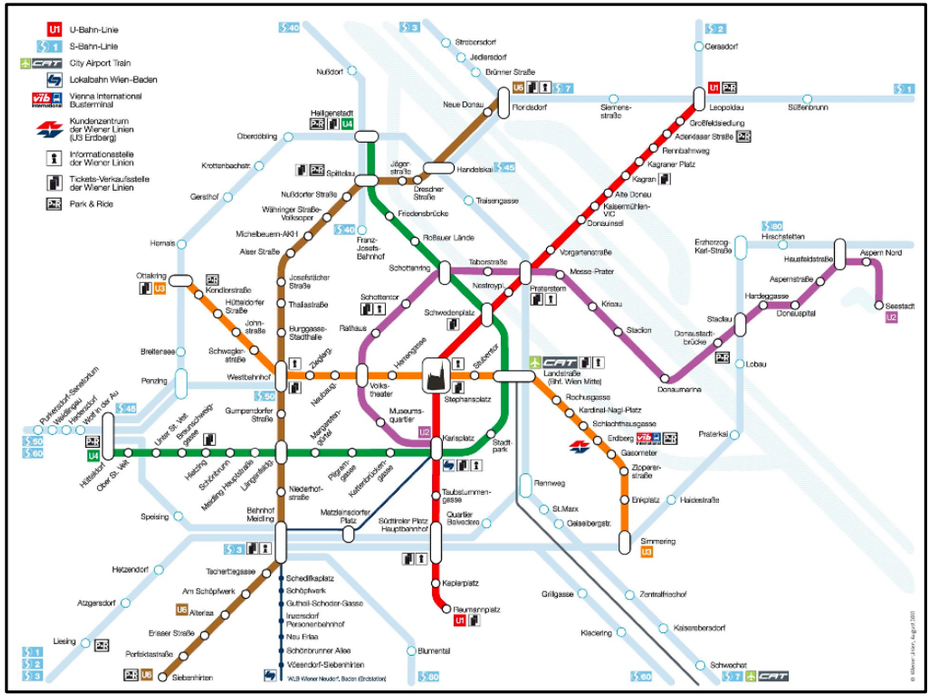

U-Bahn-Linie
S-Bahn-Linie
City Airport Train
Lokalbahn Wien–Baden
Vienna International Busterminal
Kundenzentrum der Wiener Linien (U3 Erdberg)
Informationsstelle der Wiener Linien
Tickets-Verkaufsstelle der Wiener Linien
Park & Ride
Stephansplatz
Karlsplatz
Schwedenplatz
Westbahnhof
Praterstern
Hauptbahnhof
Südtiroler Platz
Landstraße (Bhf. Wien Mitte)
Volkstheater
Museumsquartier
Rathaus
Schottentor
Schottenring
Taborstraße
Messe-Prater
Krieau
Stadion
Donaumarina
Donaustadtbrücke
Lobau
Leopoldau
Floridsdorf
Heiligenstadt
Spittelau
Friedensbrücke
Handelskai
Nußdorf
Oberdöbling
Gersthof
Michelbeuern-AKH
Alser Straße
Josefstädter Straße
Thaliastraße
Hütteldorfer Straße
Kendlerstraße
Ottakring
Hernals
Breitensee
Penzing
Hütteldorf
Ober St. Veit
Unter St. Veit
Hietzing
Schönbrunn
Meidling Hauptstraße
Längenfeldg.
Niederhofstraße
Bahnhof Meidling
Speising
Hetzendorf
Atzgersdorf
Alterlaa
Erlaaer Straße
Perfektastraße
Liesing
Siebenhirten
Tscherttegasse
Schedifkaplatz
Schöpfwerk
Am Schöpfwerk
Gutheil-Schoder-Gasse
Inzersdorf Personenbahnhof
Neu Erlaa
Schönbrunner Allee
Vösendorf-Siebenhirten
Blumental
Kaiserebersdorf
Schwechat
Simmering
Gasometer
Zentralfriedhof
Kaiserstraße
Kagran
Alte Donau
Donauinsel
Großfeldsiedlung
Rennbahnweg
Aderklaaer Straße
Süßenbrunn
Seestadt
Aspern Nord
Hausfeldstraße
Aspernstraße
Hardeggasse
Donaucity
Stadlau
Erzherzog-Karl-Straße
Hirschstetten
Prater Hauptallee
Prater Kai
Ferdinand-Nagl-Platz
Schlachthausgasse
Rochusgasse
Erdberg
Zollergasse
Herrengasse
Landstraßer Gürtel
Rennweg
St. Marx
Gaudenzdorfer Gürtel
Margaretengürtel
Pilgramgasse
Kettenbrückengasse
Taubstummengasse
Quartier Belvedere
Matzleinsdorfer Platz
Keplerplatz
Reumannplatz
Grillgasse
Kliebergasse
Gellertgasse
Zieglergasse
Zippererstraße
Ennsplatz
Haidestraße
Nestroyplatz
Jägerstraße
Dresdner Straße
Traisengasse
Roßauer Lände
Franz-Josefs-Bahnhof
Währinger Straße-Volksoper
Nußdorfer Straße
Kreuzherrengasse
Schlickgasse
Burggasse-Stadthalle
Johnstraße
Schwarzenbergplatz
Gumpendorfer Straße
Neubaugasse
Mariahilfer Straße
Zieglergasse
Stadtpark
Donaustadtstraße
Donaucity
Seestadt
Aspern Nord
Hausfeldstraße
Aspernstraße
Hardeggasse
Donauspital
Hirschstetten
Erzherzog-Karl-Straße

Hard Constraints

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

(R2) draw all edges **octilinearly**



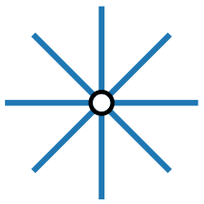


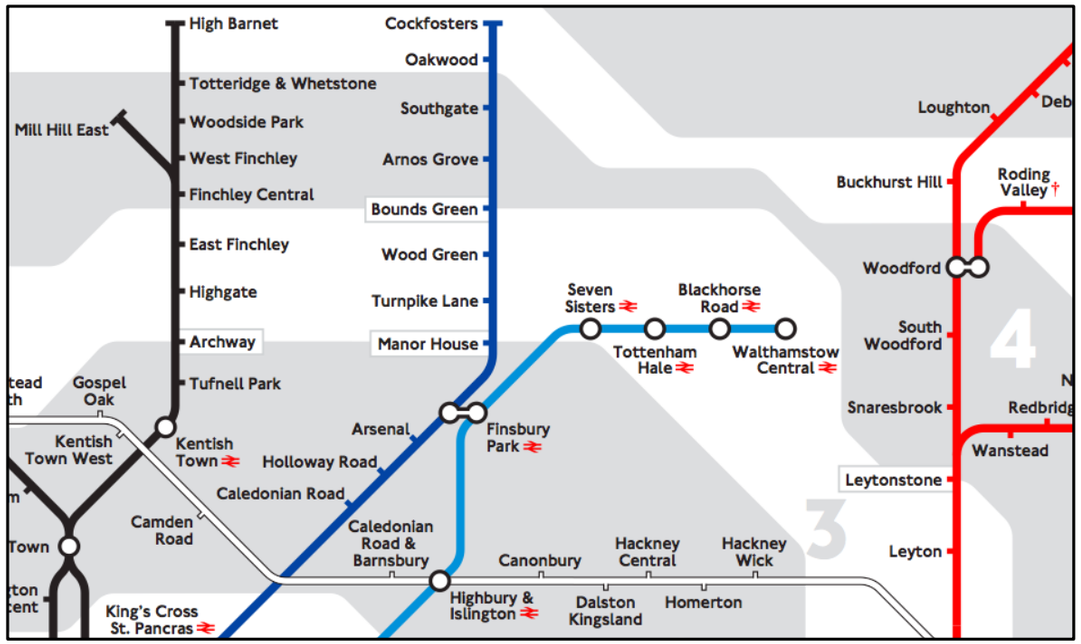
Hard Constraints

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

(R2) draw all edges **octilinearly**

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





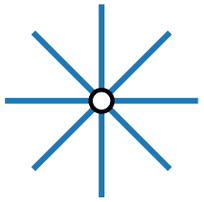
Hard Constraints

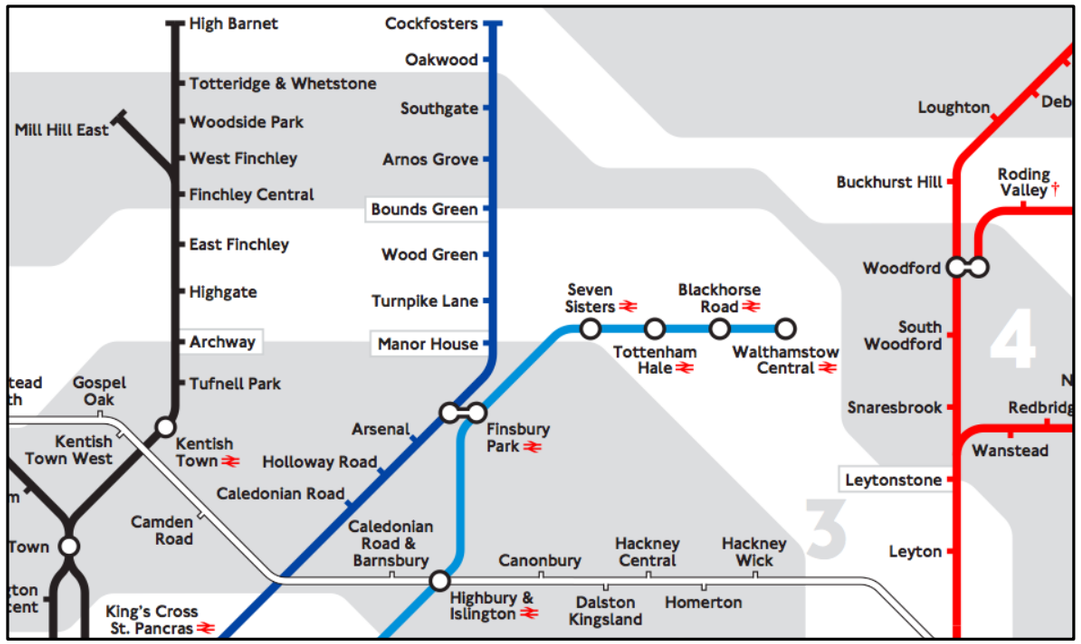
(R1) preserve embedding/**topology** and **planarity**

(R2) draw all edges **octilinearly**

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

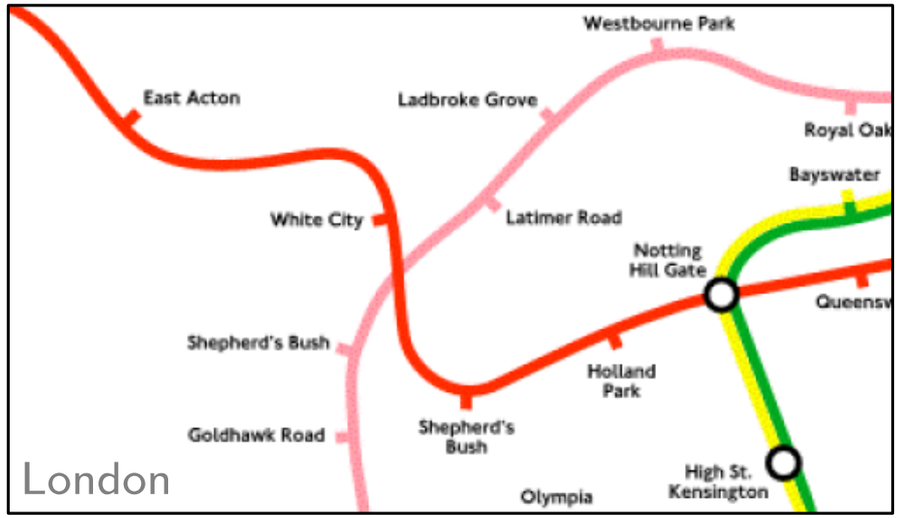
(R8) enforce **minimum feature separation** d_{min}

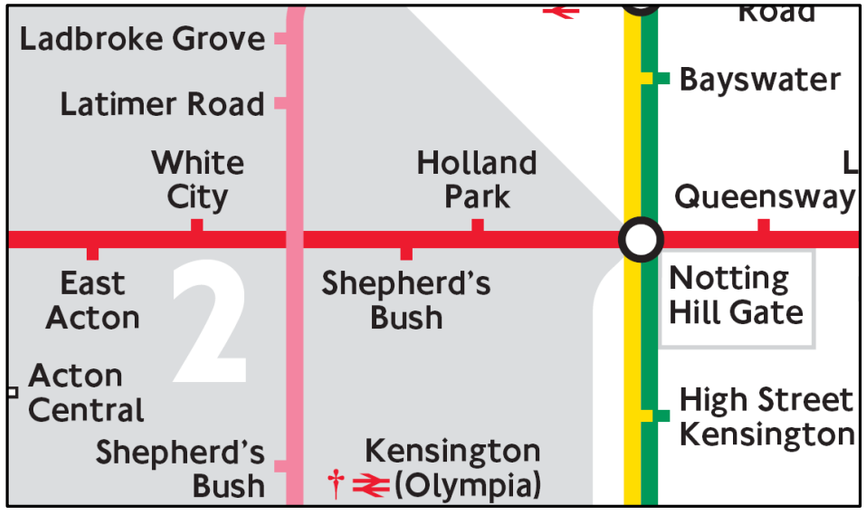




Soft Constraints

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

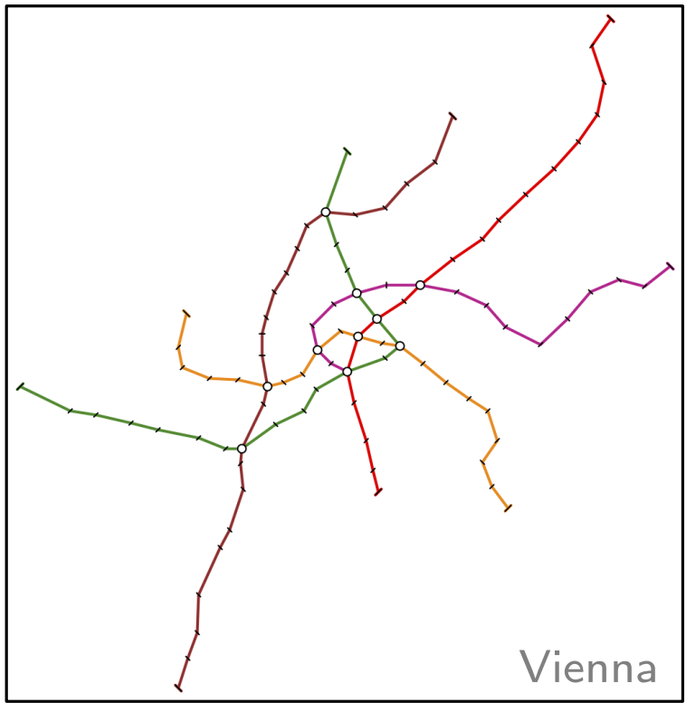


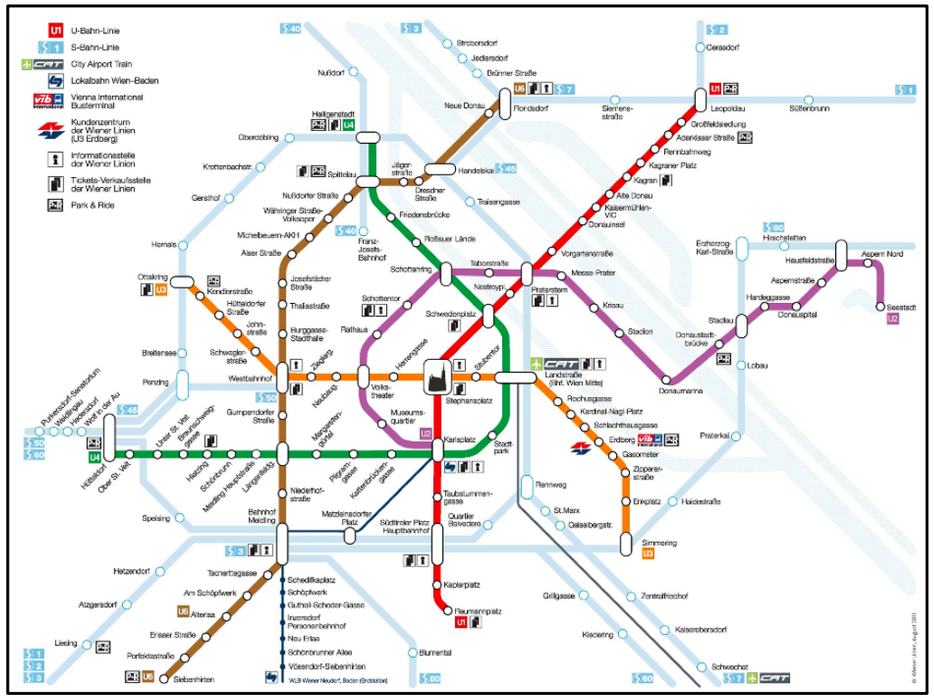


Soft Constraints

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

(R6) minimize geometric **distortion**



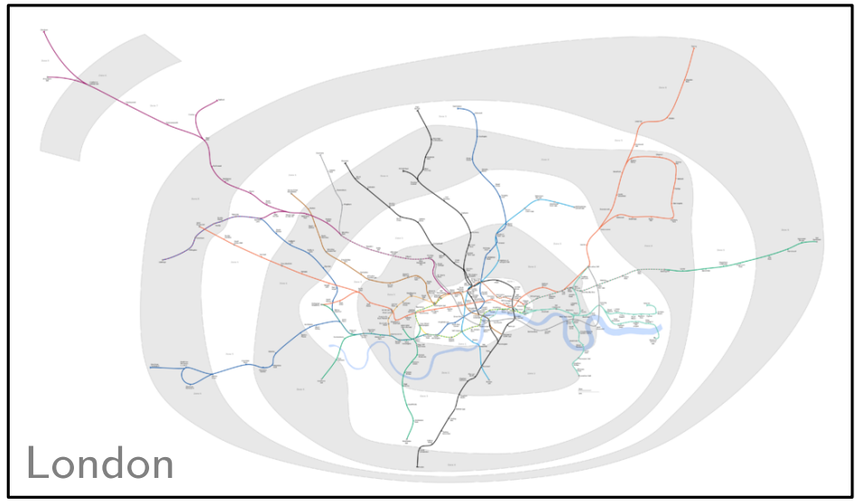


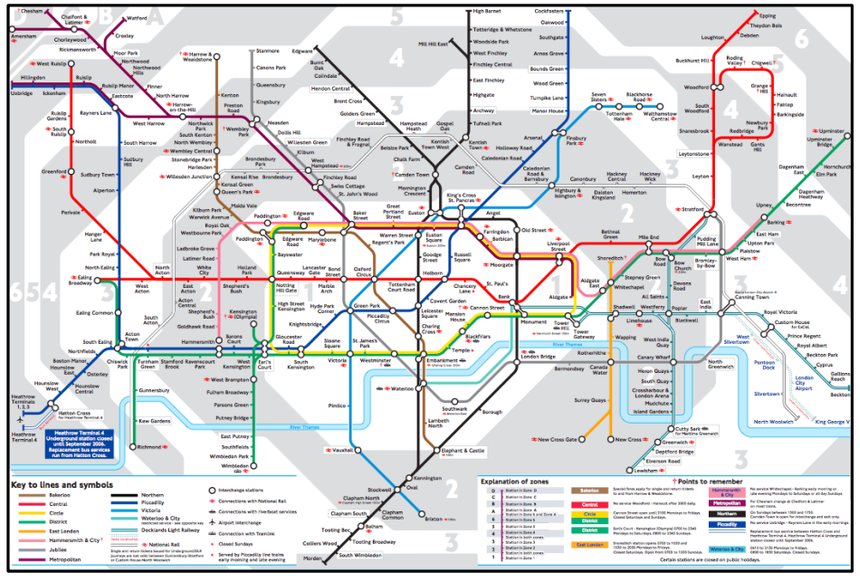
Soft Constraints

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

(R6) minimize geometric **distortion**

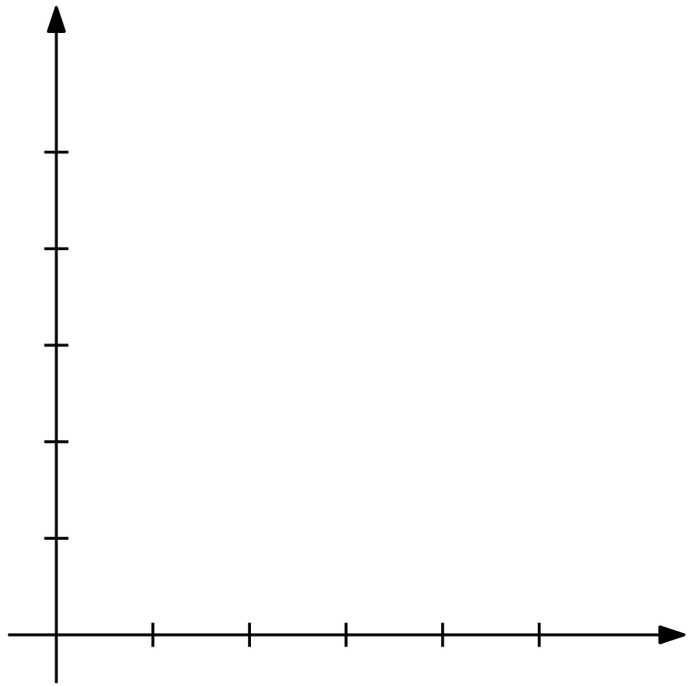
(R7) minimize total **edge length**





Linear Programming

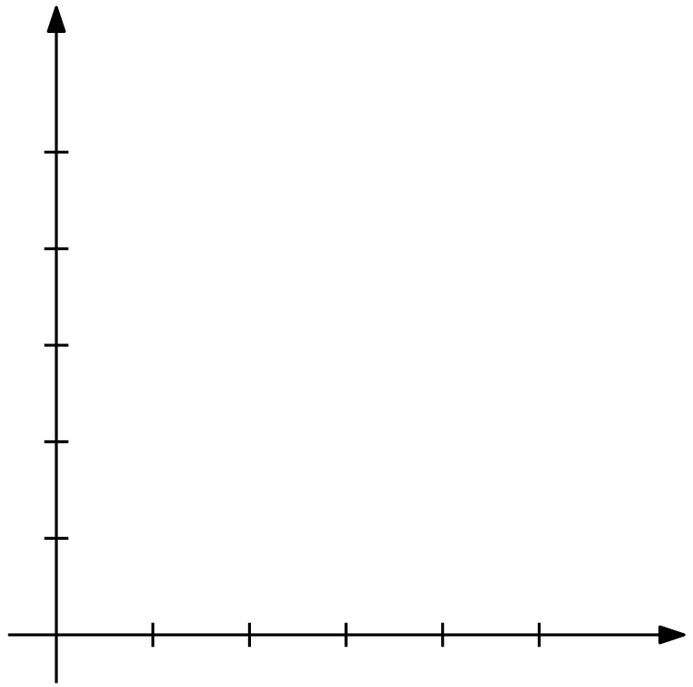
Linear Programming (LP) is an efficient optimization method for



Linear Programming

Linear Programming (LP) is an efficient optimization method for

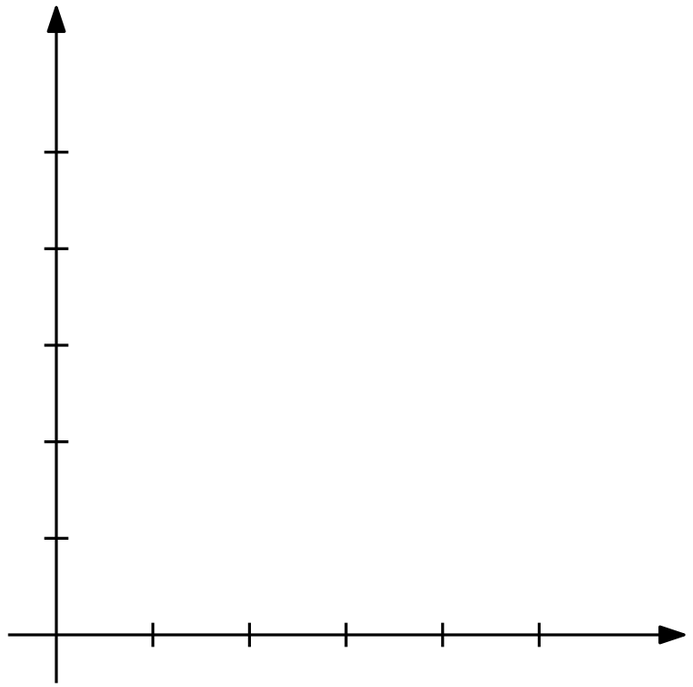
- linear constraints



Linear Programming

Linear Programming (LP) is an efficient optimization method for

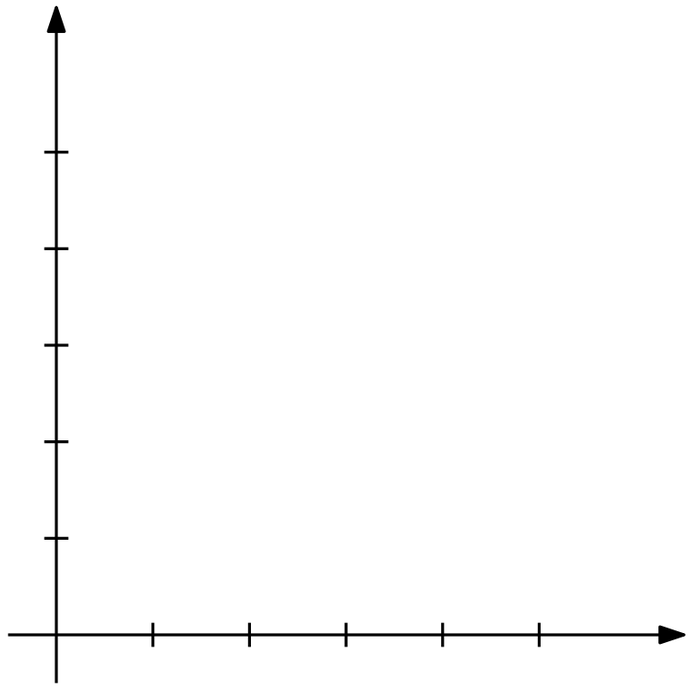
- linear constraints
- linear objective function



Linear Programming

Linear Programming (LP) is an efficient optimization method for

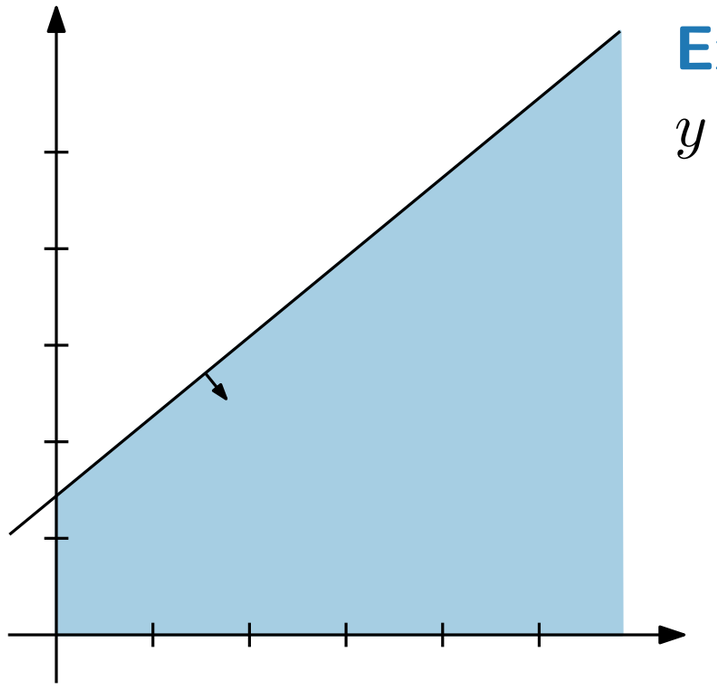
- linear constraints
- linear objective function
- real-valued variables



Linear Programming

Linear Programming (LP) is an efficient optimization method for

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



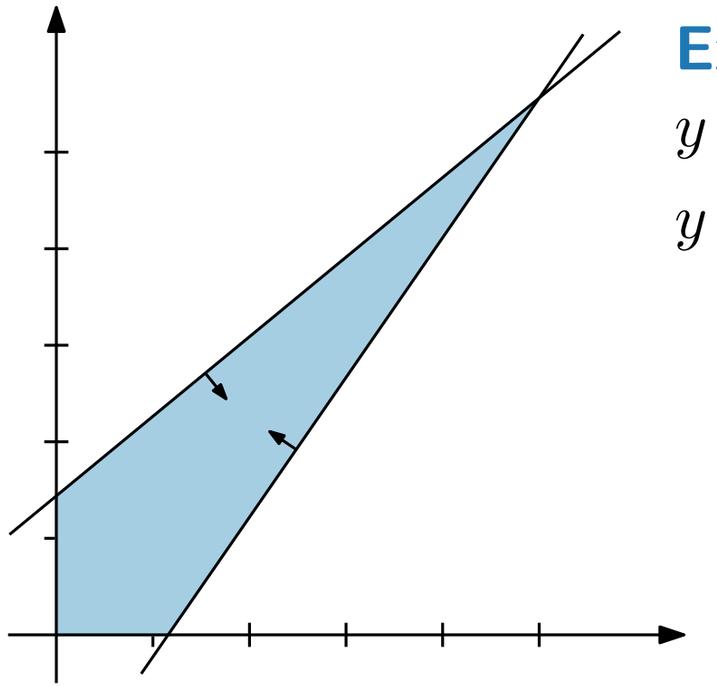
Example.

$y \leq 0.9x + 1.5$

Linear Programming

Linear Programming (LP) is an efficient optimization method for

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



Example.

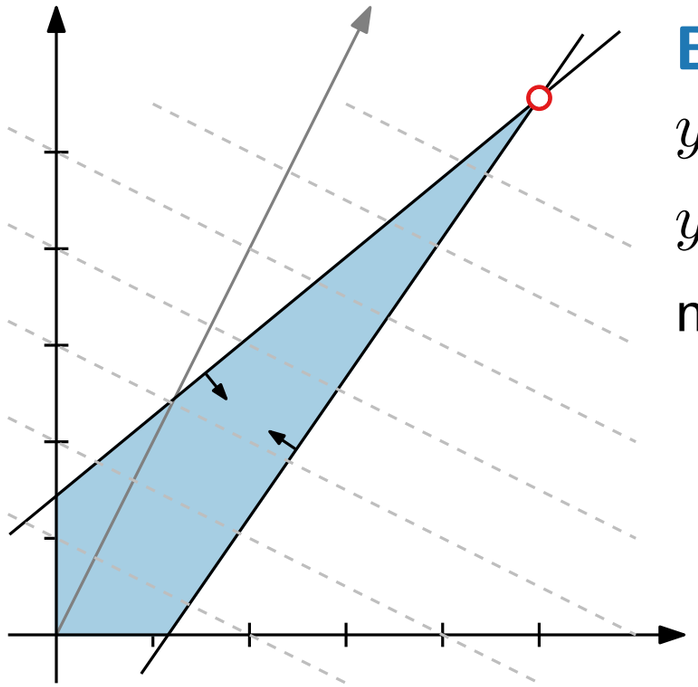
$y \leq 0.9x + 1.5$

$y \geq 1.4x - 1.3$

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 $x + 2y$

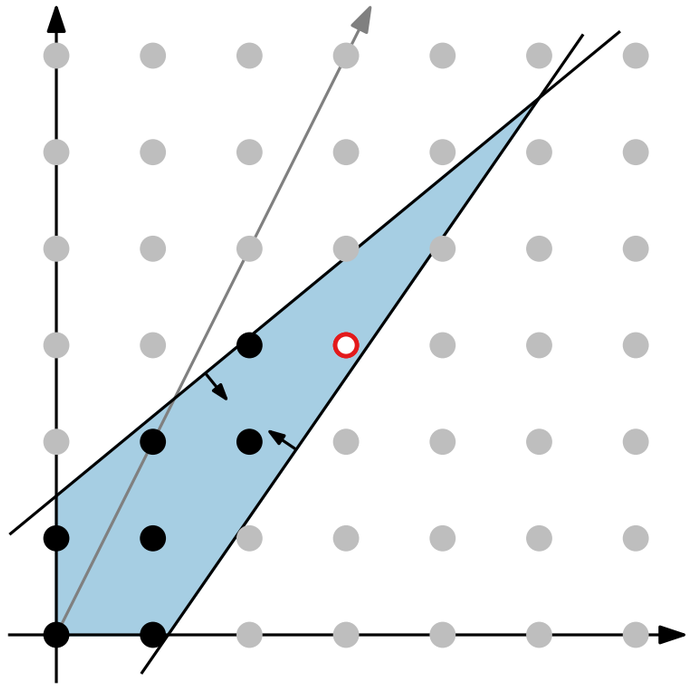
Linear Programming

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



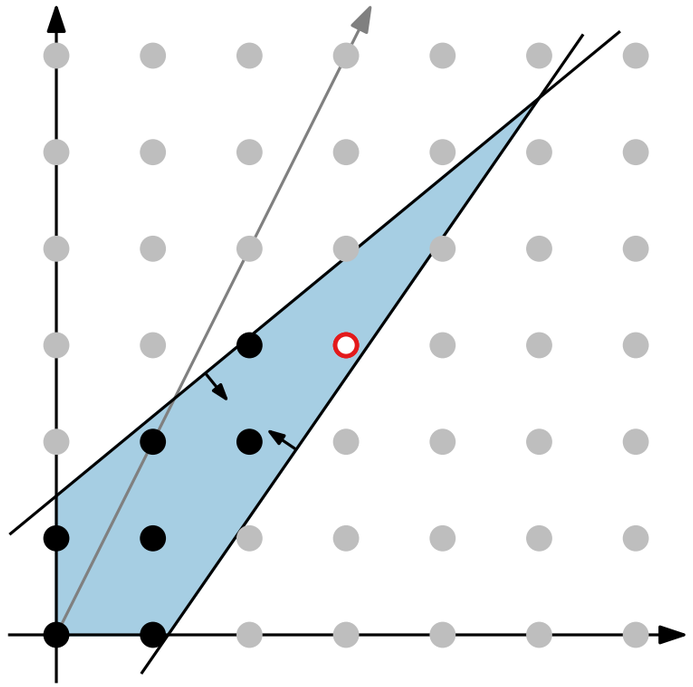
Linear Programming

Linear Programming (LP) is an efficient optimization method for

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

Mixed Integer Programming (MIP)

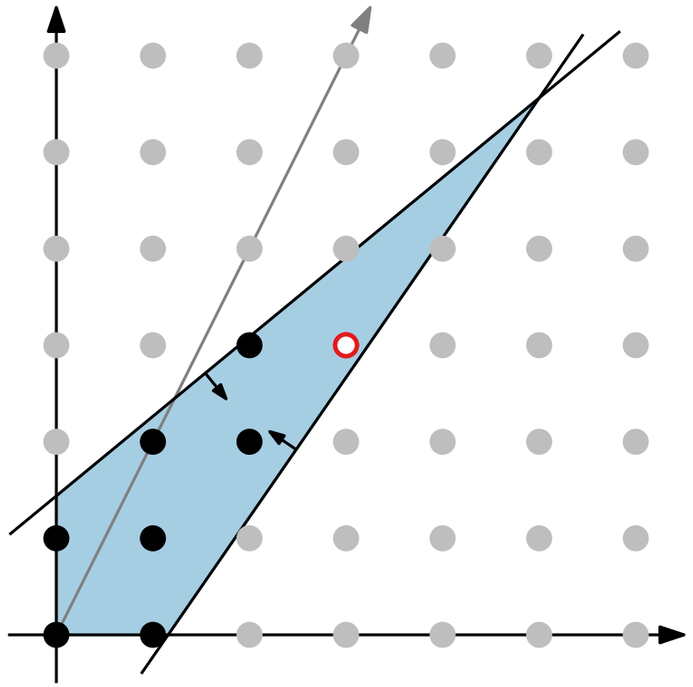
- in addition binary and integer variables
- NP-hard optimization problem



Linear Programming

Linear Programming (LP) is an efficient optimization method for

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



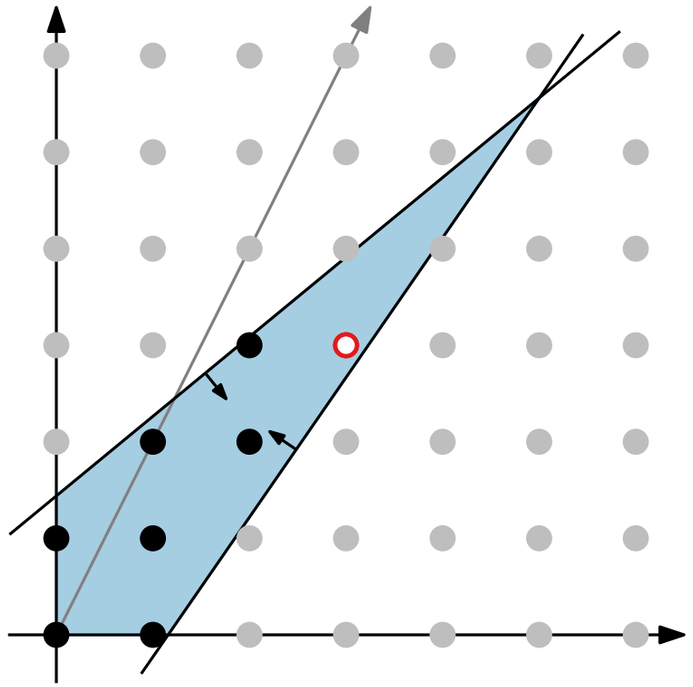
Mixed Integer Programming (MIP)

- 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



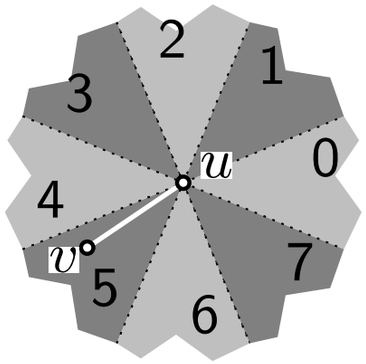
Mixed Integer Programming (MIP)

- 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

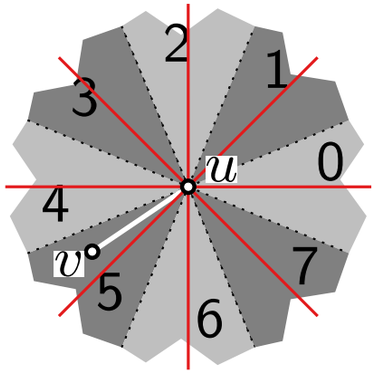
Sectors and Coordinates



Sectors.

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

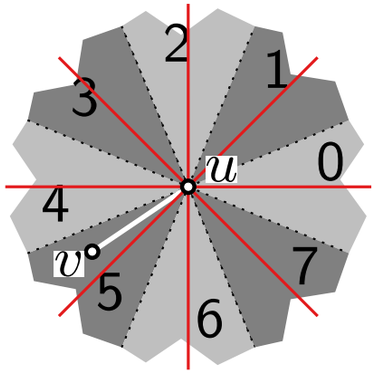
Sectors and Coordinates



Sectors.

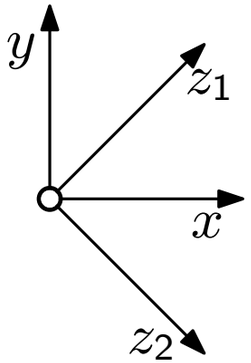
- for each vertex u partition the plane into eight sectors numbered 0–7
  here: $\text{sec}(u, v) = 5$ in the input
- number octilinear edge directions accordingly
  here, e.g., $\text{dir}(u, v) = 5$ in the solution (output)

Sectors and Coordinates



Sectors.

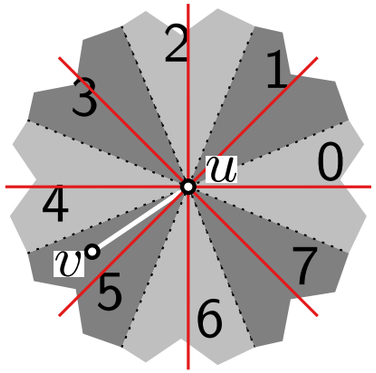
- for each vertex u partition the plane into eight sectors numbered 0–7
  here: $\text{sec}(u, v) = 5$ in the input
- number octilinear edge directions accordingly
  here, e.g., $\text{dir}(u, v) = 5$ in the solution (output)



Coordinates.

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

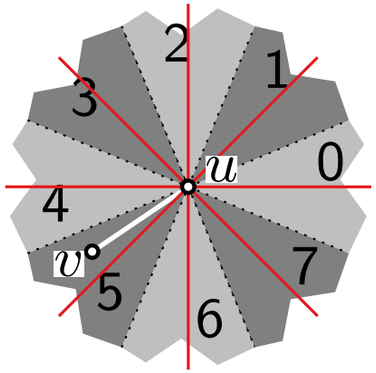
Octilinearity and Relative Positions



Goal.

Draw the edge uv

Octilinearity and Relative Positions

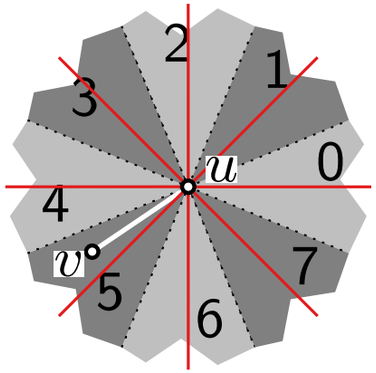


Goal.

Draw the edge uv

- octilinearly **(R2)**

Octilinearity and Relative Positions

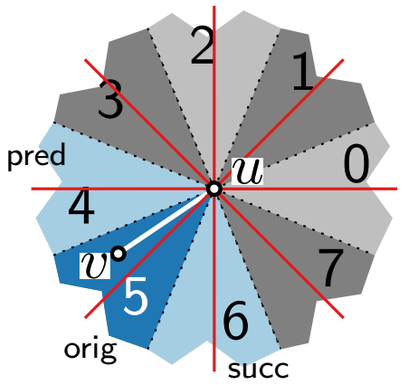


Goal.

Draw the edge uv

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

Octilinearity and Relative Positions

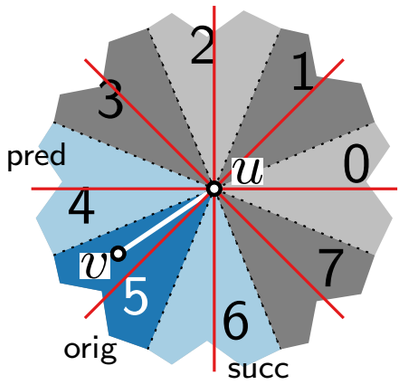


Goal.

Draw the edge uv

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

Octilinearity and Relative Positions



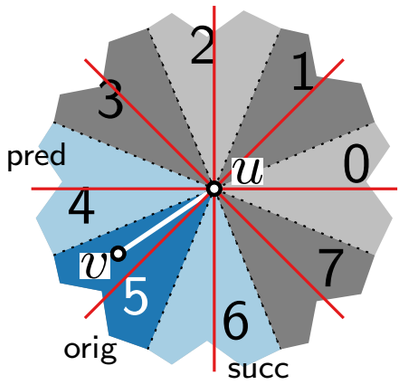
Goal.

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 Positions



Goal.

Draw the edge uv

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

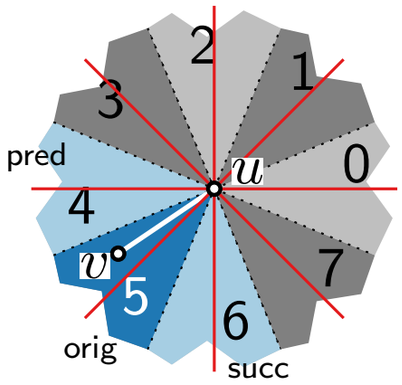
How to model this using linear constraints in a MIP?

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 Positions



Goal.

Draw the edge uv

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

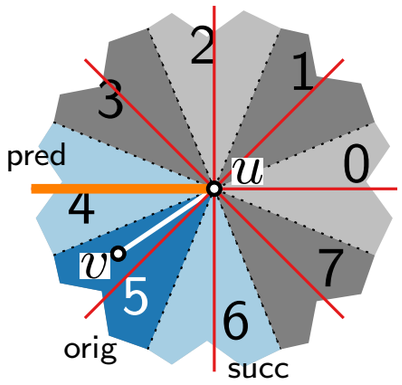
How to model this using linear constraints in a MIP?

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.

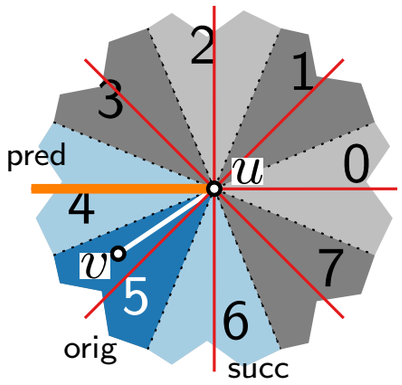
Octilinearity and Relative Positions



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 Positions

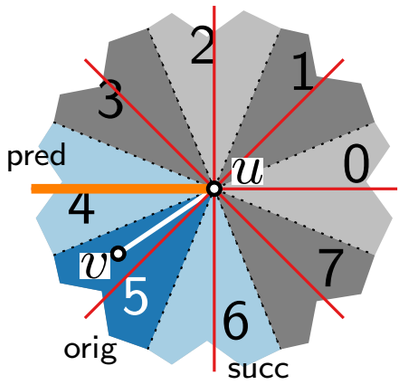


Predecessor sector.

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 Positions



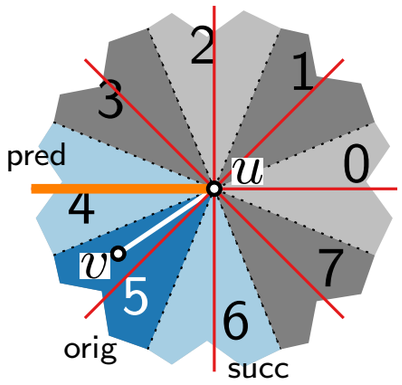
Predecessor sector.

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 Positions



Predecessor sector.

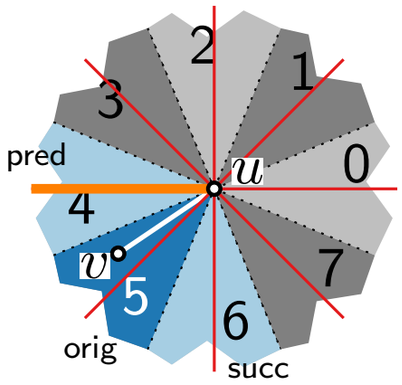
very large constant

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

How does this work?

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

Octilinearity and Relative Positions



Predecessor sector.

very large constant

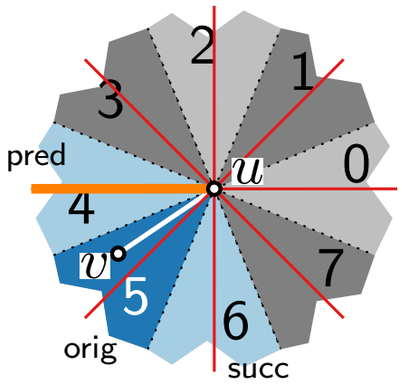
$$\begin{aligned} y(u) - y(v) &\leq M(1 - \alpha_{\text{pred}}(u,v)) \\ -y(u) + y(v) &\leq M(1 - \alpha_{\text{pred}}(u,v)) \\ x(u) - x(v) &\geq -M(1 - \alpha_{\text{pred}}(u,v)) + \ell \end{aligned}$$

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 Positions



Predecessor sector.

very large constant

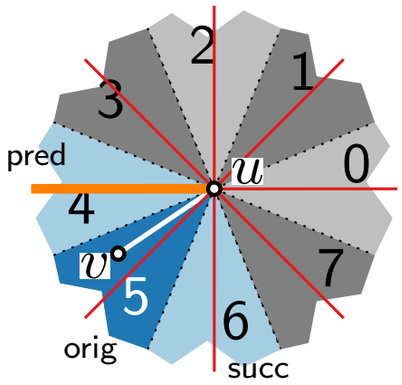
$$\begin{aligned} y(u) - y(v) &\leq M(1 - \alpha_{\text{pred}}(u,v)) \\ -y(u) + y(v) &\leq M(1 - \alpha_{\text{pred}}(u,v)) \\ x(u) - x(v) &\geq -M(1 - \alpha_{\text{pred}}(u,v)) + \ell \end{aligned}$$

How does this work?

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

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

Octilinearity and Relative Positions



Predecessor sector.

very large constant

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

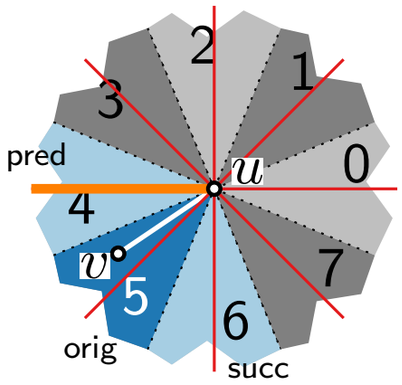
How does this work?

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

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

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

Octilinearity and Relative Positions



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?

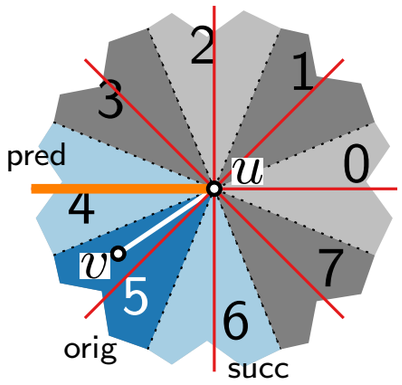
Case 1: $\alpha_{\text{pred}}(u,v) = 1$

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

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 Positions



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?

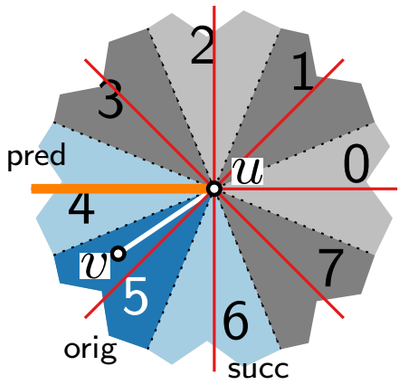
Case 1: $\alpha_{\text{pred}}(u,v) = 1$

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

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

$$\left.\begin{aligned} y(u) - y(v) &\leq M \\ -y(u) + y(v) &\leq M \\ x(u) - x(v) &\geq -M + \ell \end{aligned}\right\} \text{all arbitrary}$$

Octilinearity and Relative Positions



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?

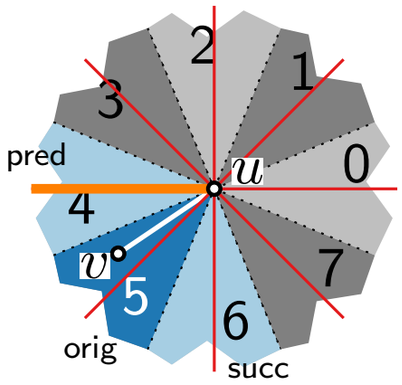
Case 1: $\alpha_{\text{pred}}(u,v) = 1$

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

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

$$\left.\begin{aligned} y(u) - y(v) &\leq M \\ -y(u) + y(v) &\leq M \\ x(u) - x(v) &\geq -M + \ell \end{aligned}\right\} \text{all arbitrary}$$

Octilinearity and Relative Positions



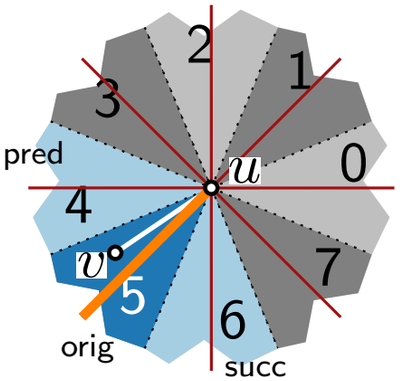
very large constant

Predecessor sector.

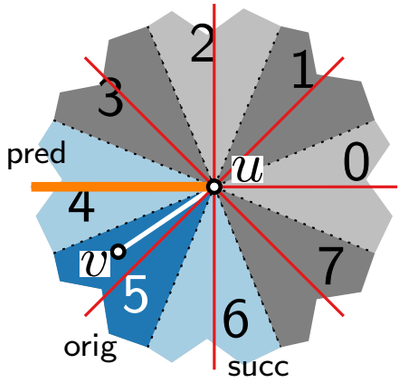
$$
\begin{aligned}
y(u) - y(v) &\leq M(1 - \alpha_{\text{pred}}(u,v)) \\
-y(u) + y(v) &\leq M(1 - \alpha_{\text{pred}}(u,v)) \\
x(u) - x(v) &\geq -M(1 - \alpha_{\text{pred}}(u,v)) + \ell
\end{aligned}
$$

Original sector.

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



Octilinearity and Relative Positions



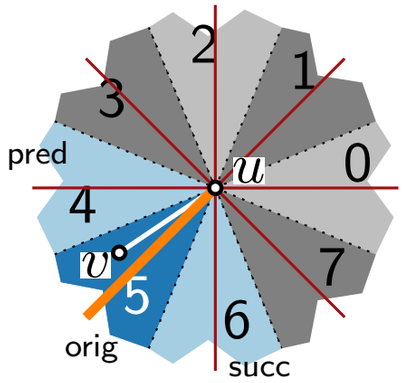
Predecessor sector.

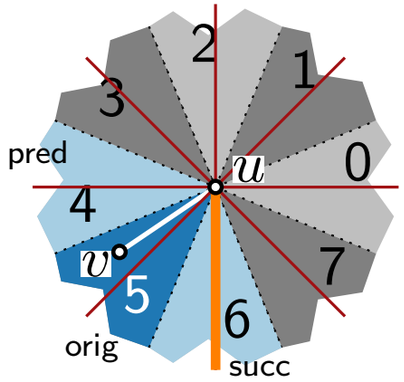
very large constant

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

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

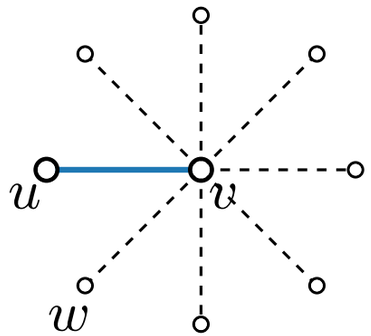
  minimize $\lambda_{\text{bends}}\,\text{cost}_{\text{bends}} + \lambda_{\text{length}}\,\text{cost}_{\text{length}} + \lambda_{\text{dist}}\text{cost}_{\text{dist}}$

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)



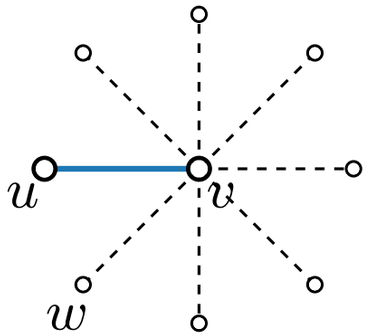
Objective Function

- 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 $L \in \mathcal{L}$

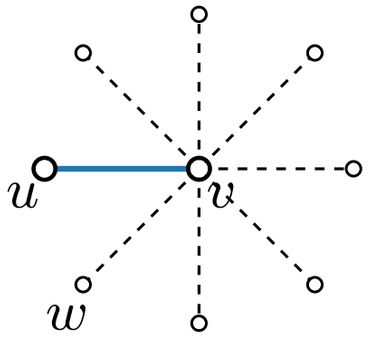


Objective Function

- 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 $L \in \mathcal{L}$

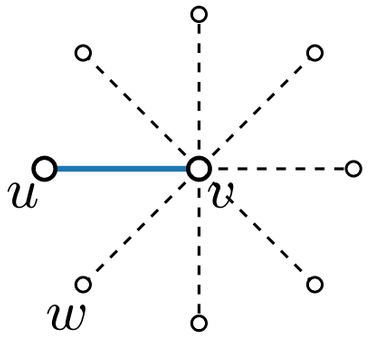
- draw as straight as possible

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 $\angle(\overline{uv}, \overline{vw})$

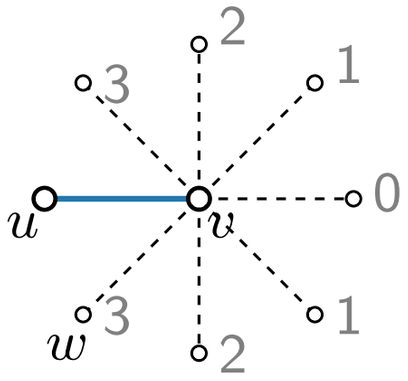
$$\text{cost}_{\text{bends}} = \sum_{L \in \mathcal{L}} \sum_{uv, vw \in L} \text{bend}(u, v, w)$$

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 $\angle(\overline{uv}, \overline{vw})$

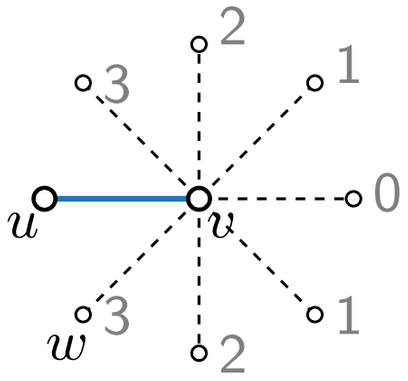
$$\text{cost}_{\text{bends}} = \sum_{L \in \mathcal{L}} \sum_{uv, vw \in L} \text{bend}(u, v, w)$$

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 $\lambda_{\text{bend}}\text{cost}_{\text{bend}} + \lambda_{\text{len}}\text{cost}_{\text{len}} + \lambda_{\text{dist}}\text{cost}_{\text{dist}}$

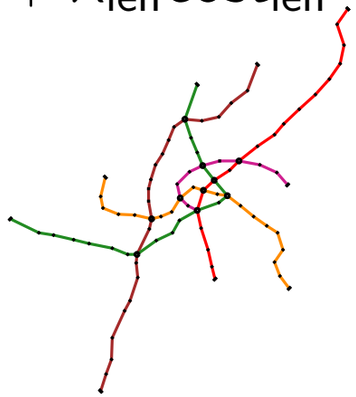
Overview MIP model

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:



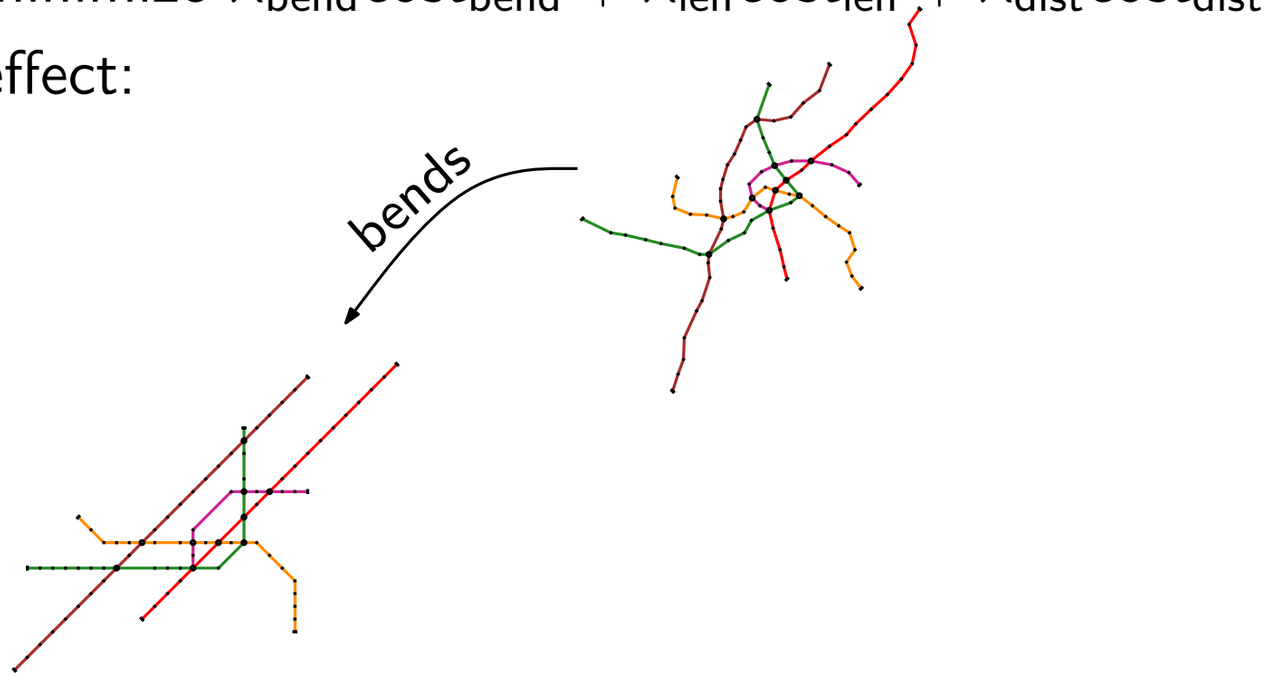
Overview MIP model

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:



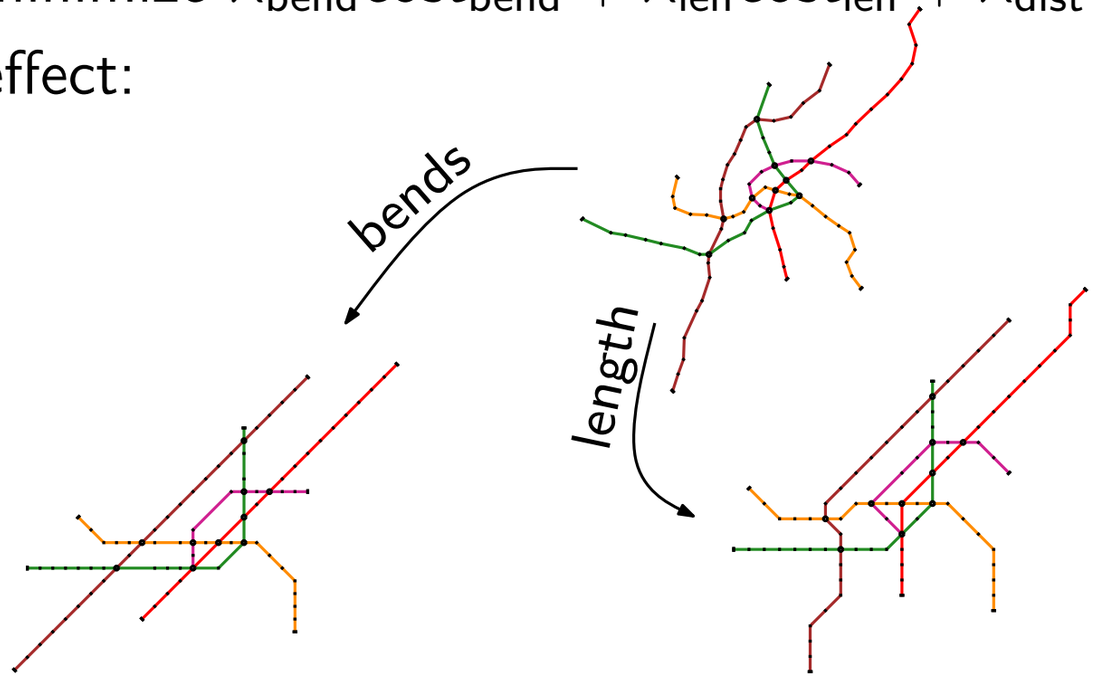
Overview MIP model

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:



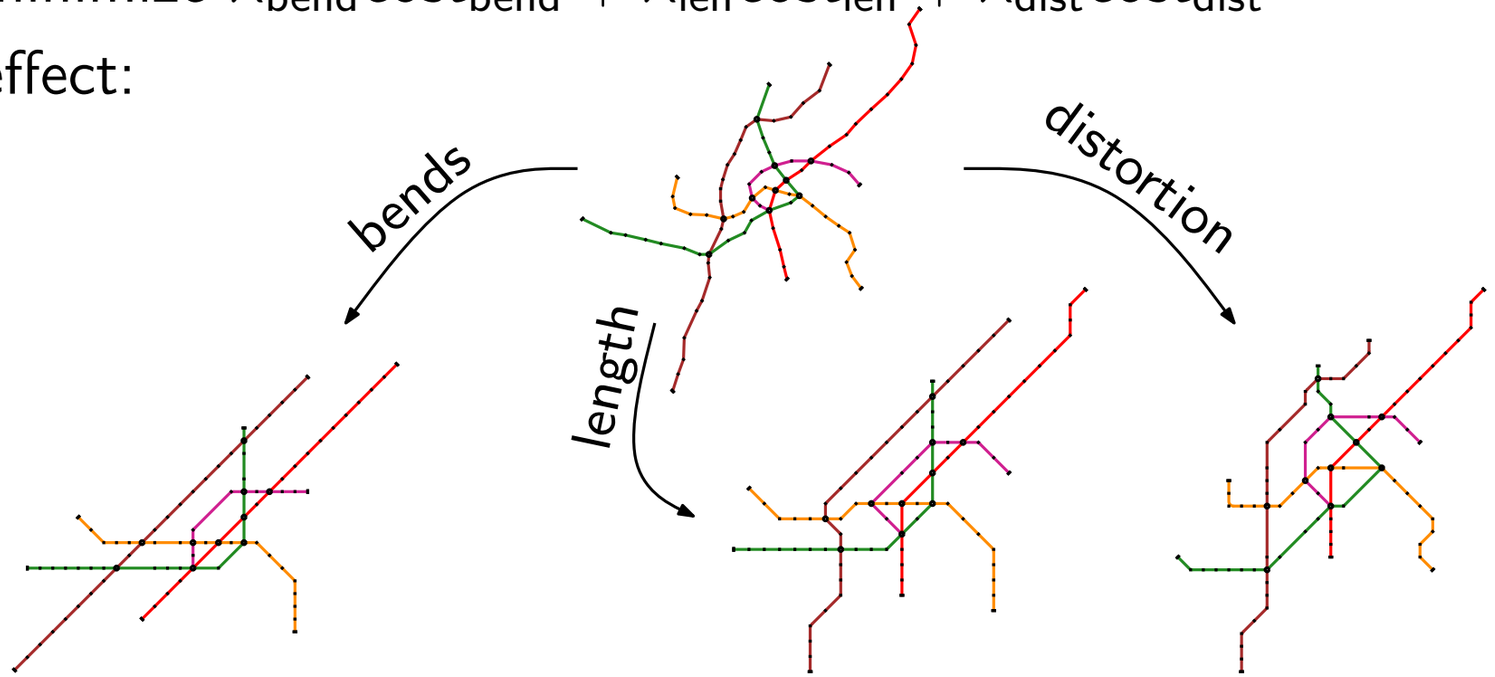
Overview MIP model

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:



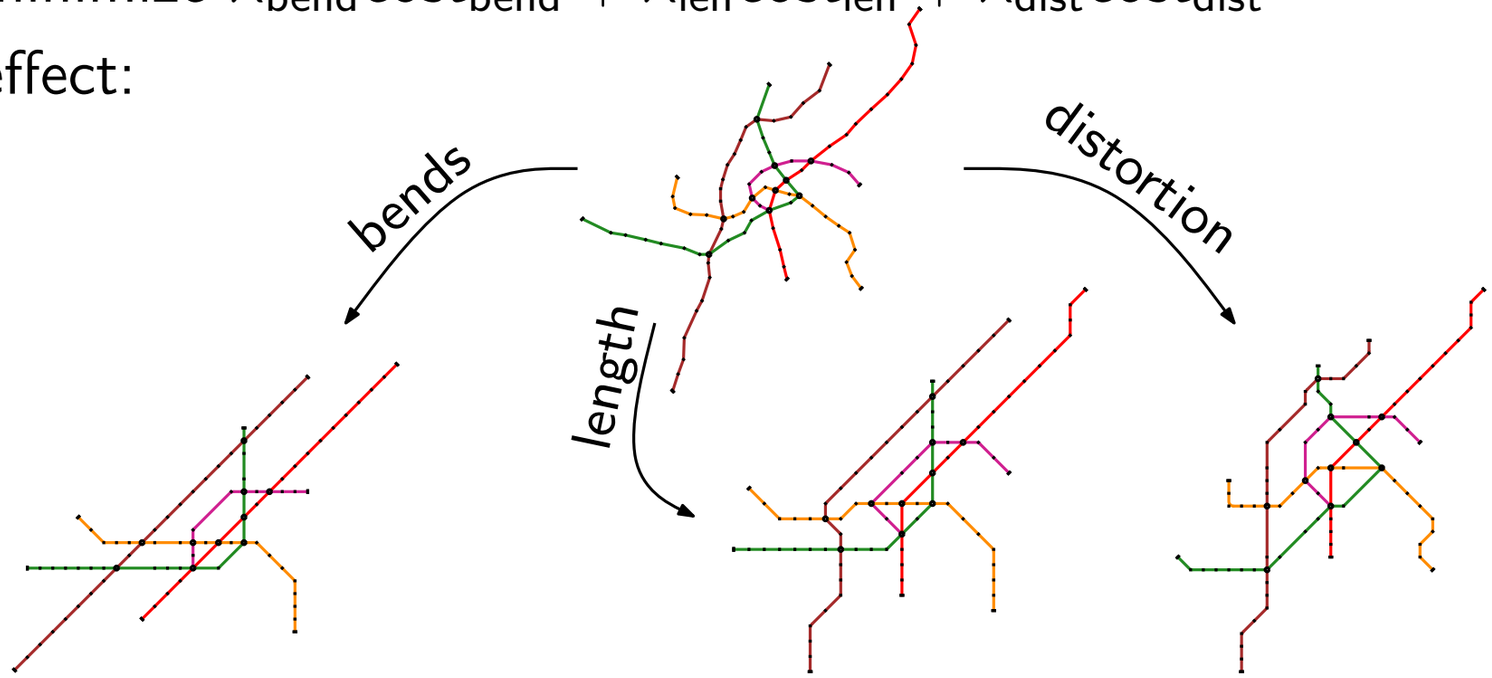
Overview MIP model

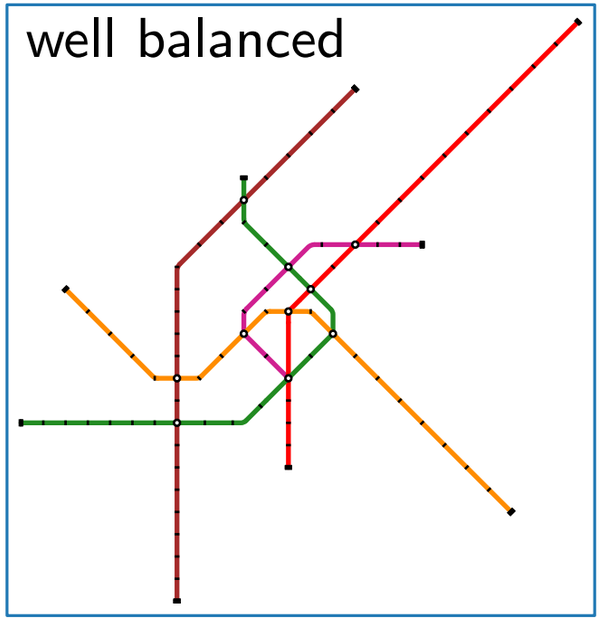
Constraints.

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

Objective function.

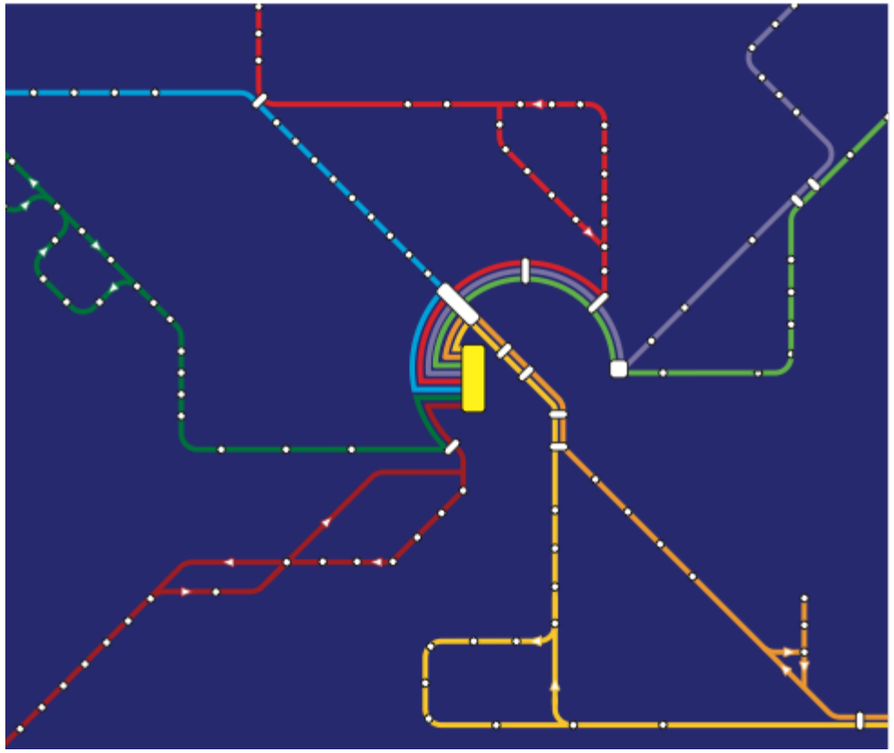
- weighted sum of the three soft constraints
- minimize $\lambda_{\text{bend}}\text{cost}_{\text{bend}} + \lambda_{\text{len}}\text{cost}_{\text{len}} + \lambda_{\text{dist}}\text{cost}_{\text{dist}}$
- effect:





Station Labeling

- unlabeled map mostly useless



Station Labeling

- unlabeled map mostly useless



Station Labeling

- unlabeled map mostly useless



Paderborn

Station Labeling

- unlabeled map mostly useless
- labels need space



Paderborn

Station Labeling

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



Paderborn

Station Labeling

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



Paderborn

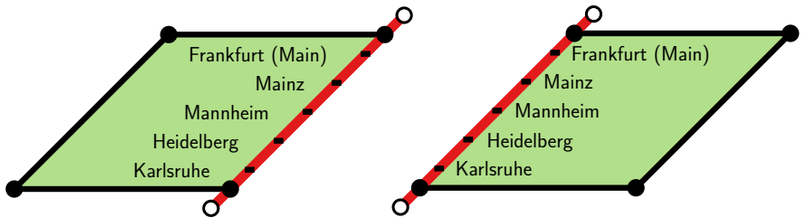
Station Labeling

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



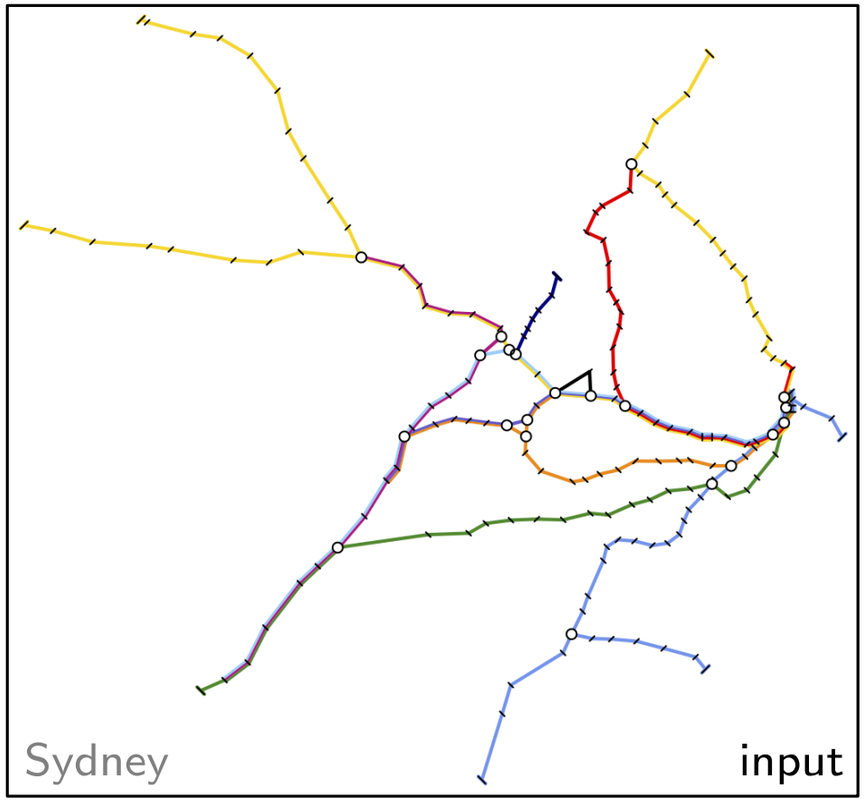
Paderborn

→ combine layout & labeling for optimal results!

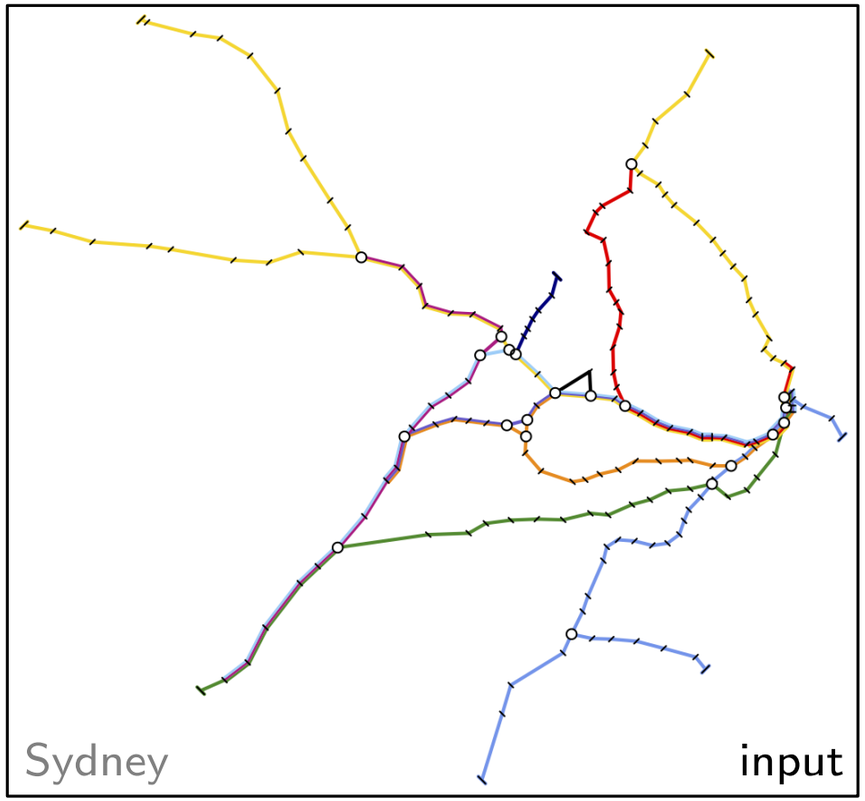


- parallelogram as special metro line
- switching sides allowed

Example: Sydney

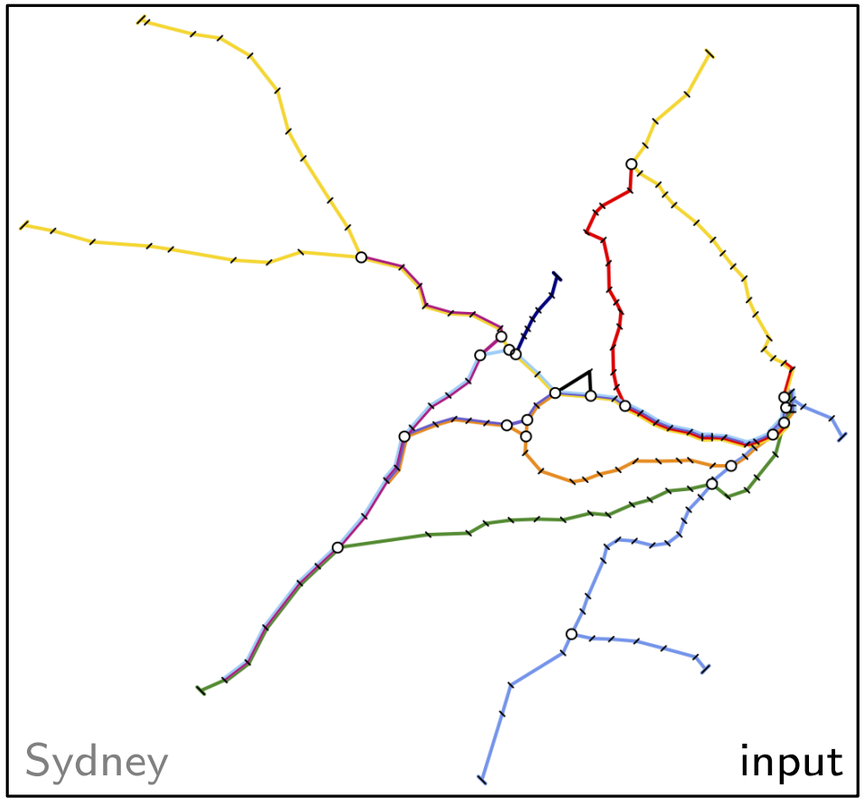


Example: Sydney



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

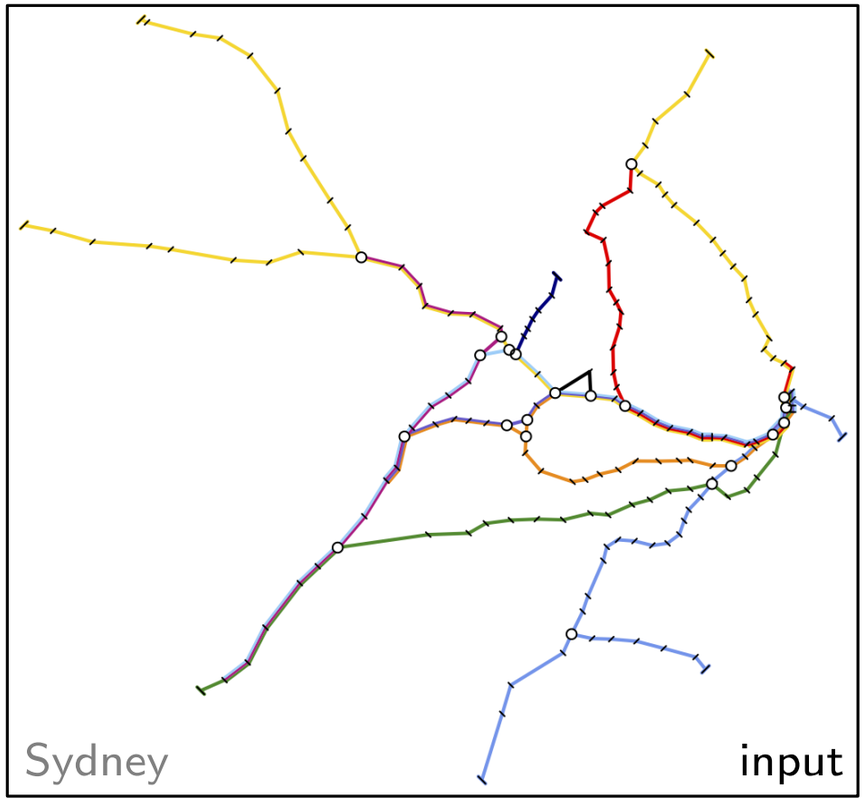
Example: Sydney



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

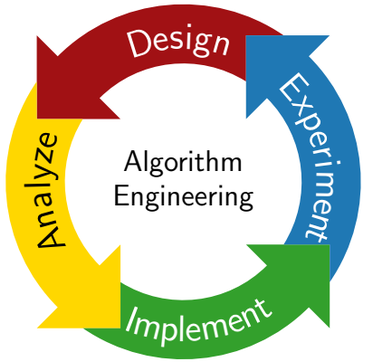
MIP	constraints	variables
full	1,191,406	290,137
callback	21,988	92,681

Example: Sydney

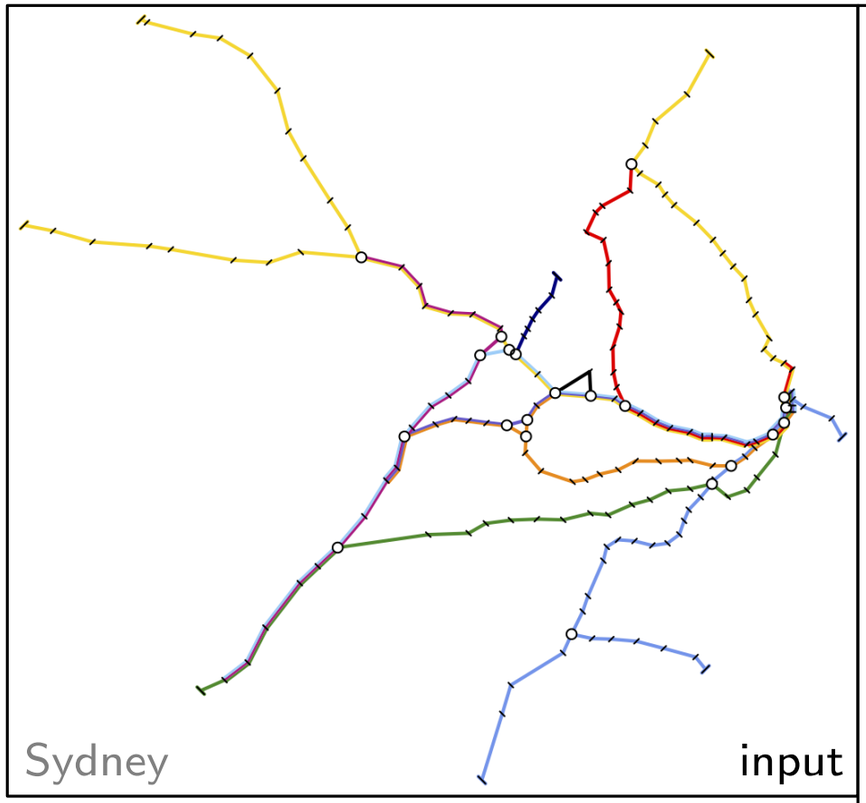


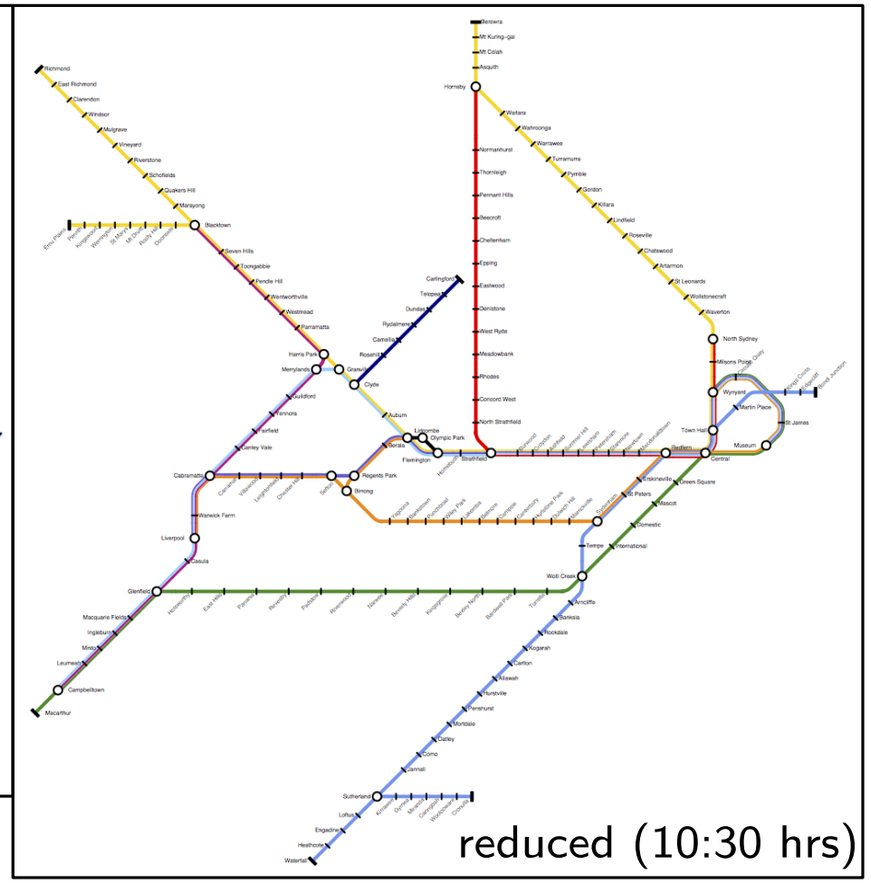
Input	$\lvert V\rvert$	$\lvert E\rvert$	faces	$\lvert\mathcal{L}\rvert$
full	174	183	11	10
reduced	88	97		
labeled	242	270	30	

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

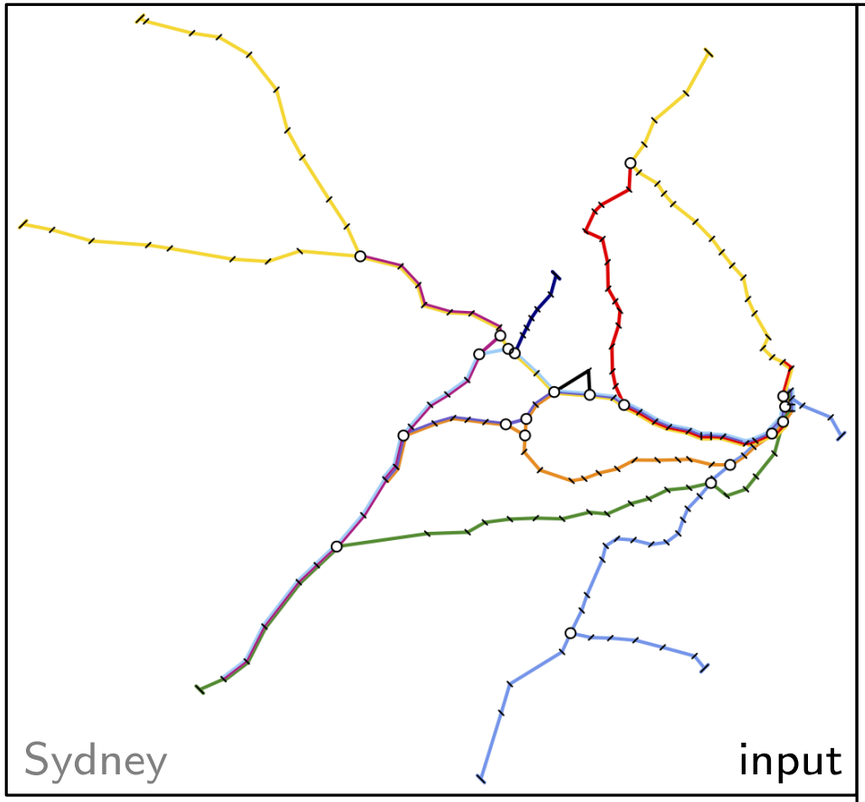


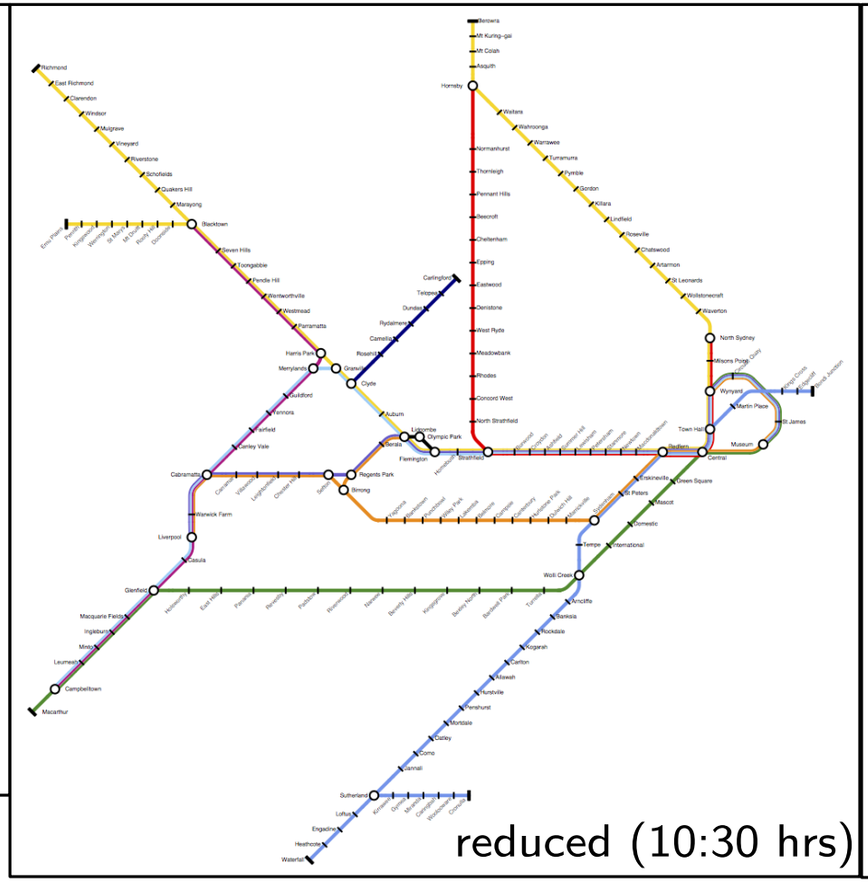
Example: Sydney

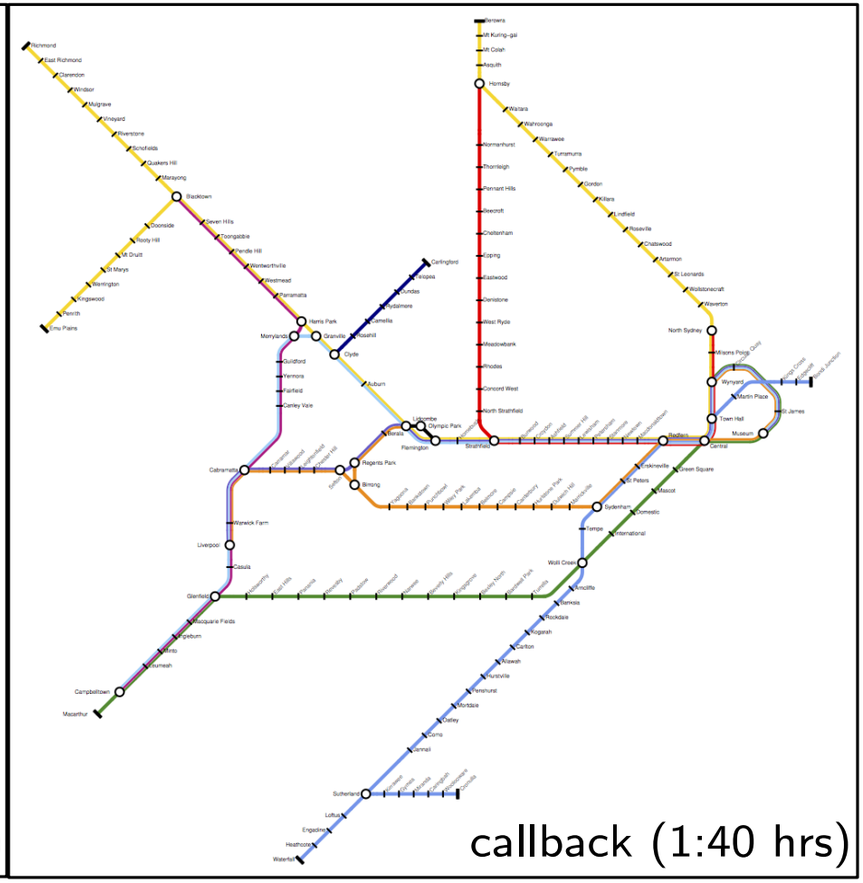




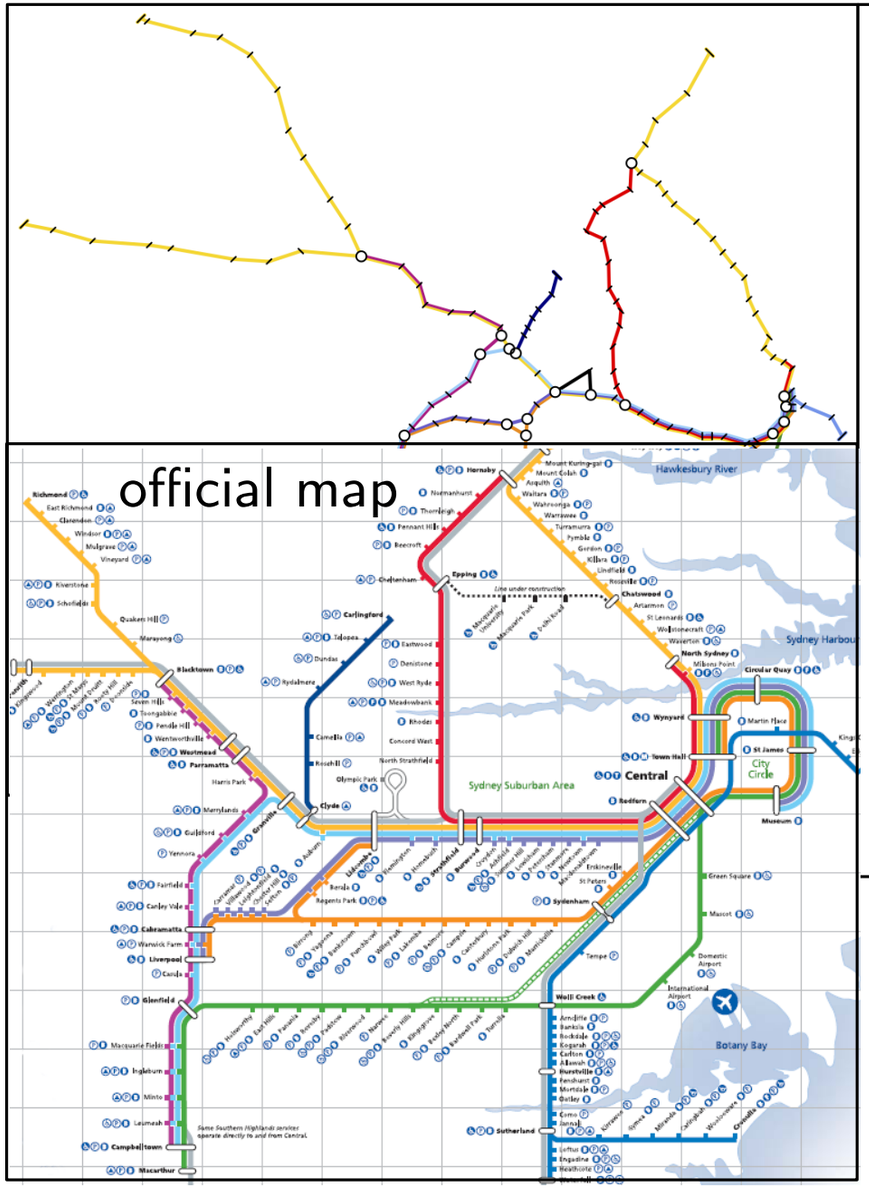
Example: Sydney

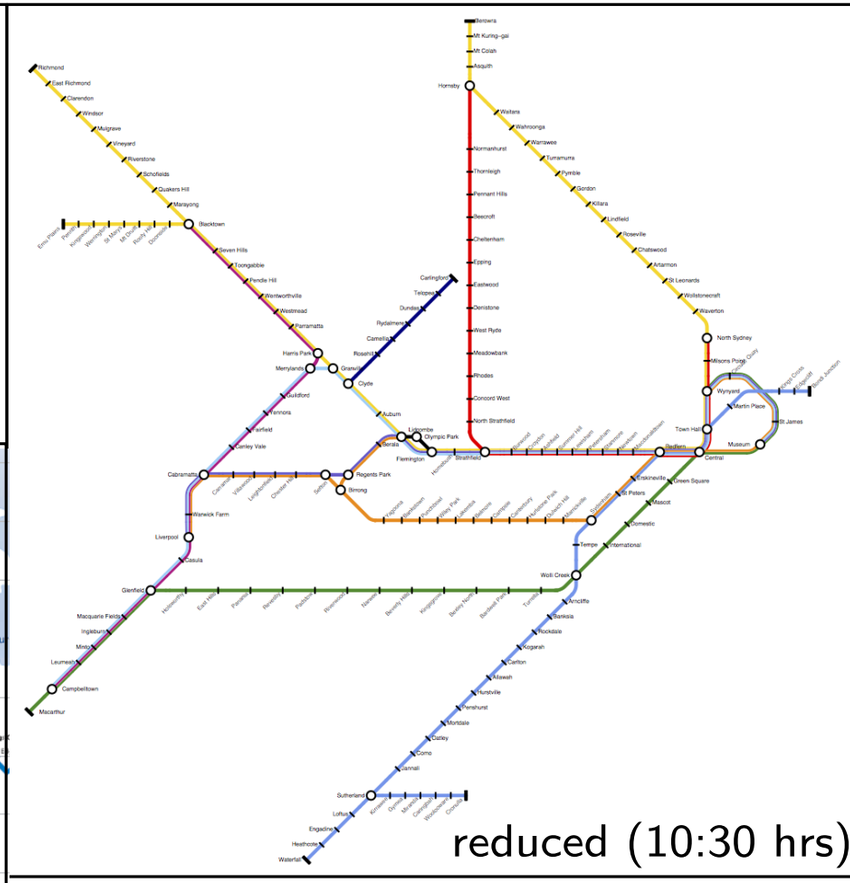


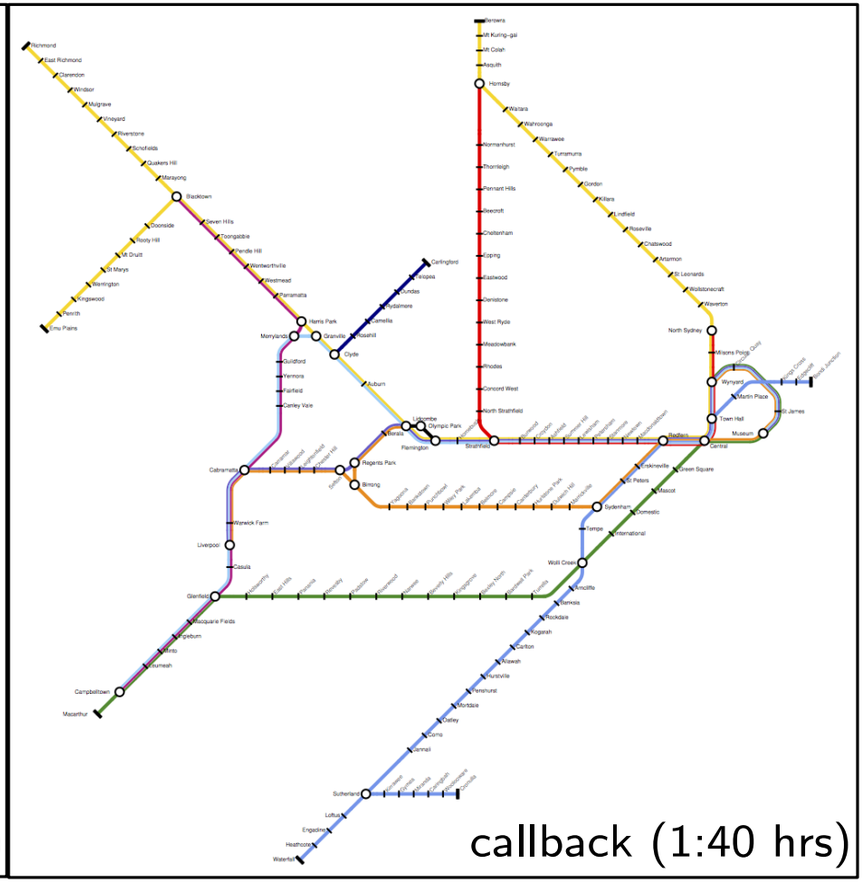




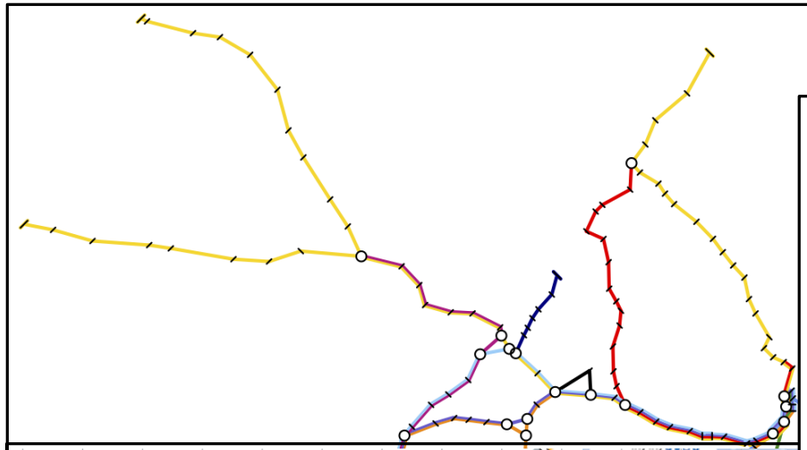
Example: Sydney



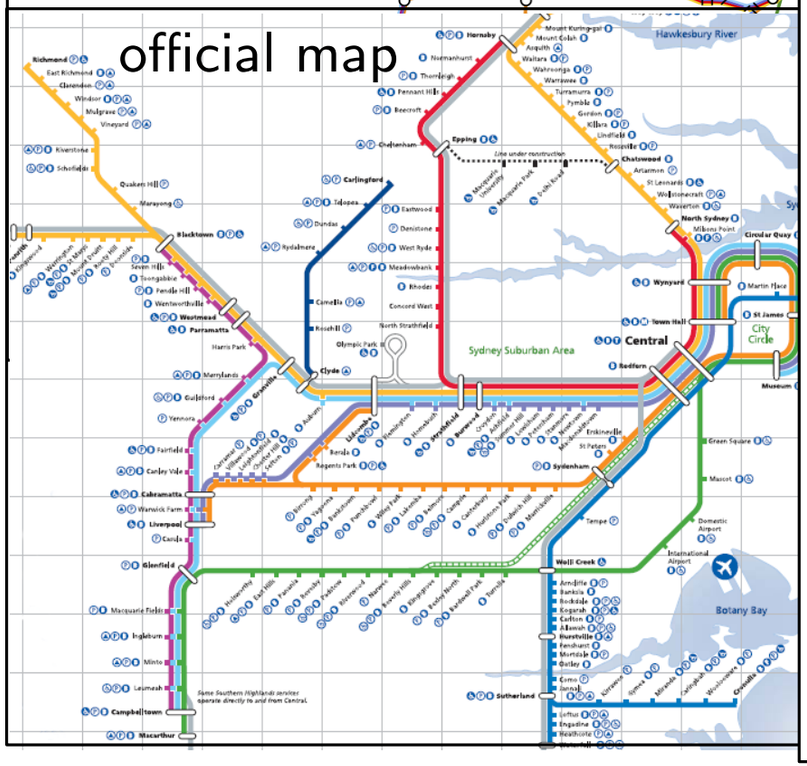




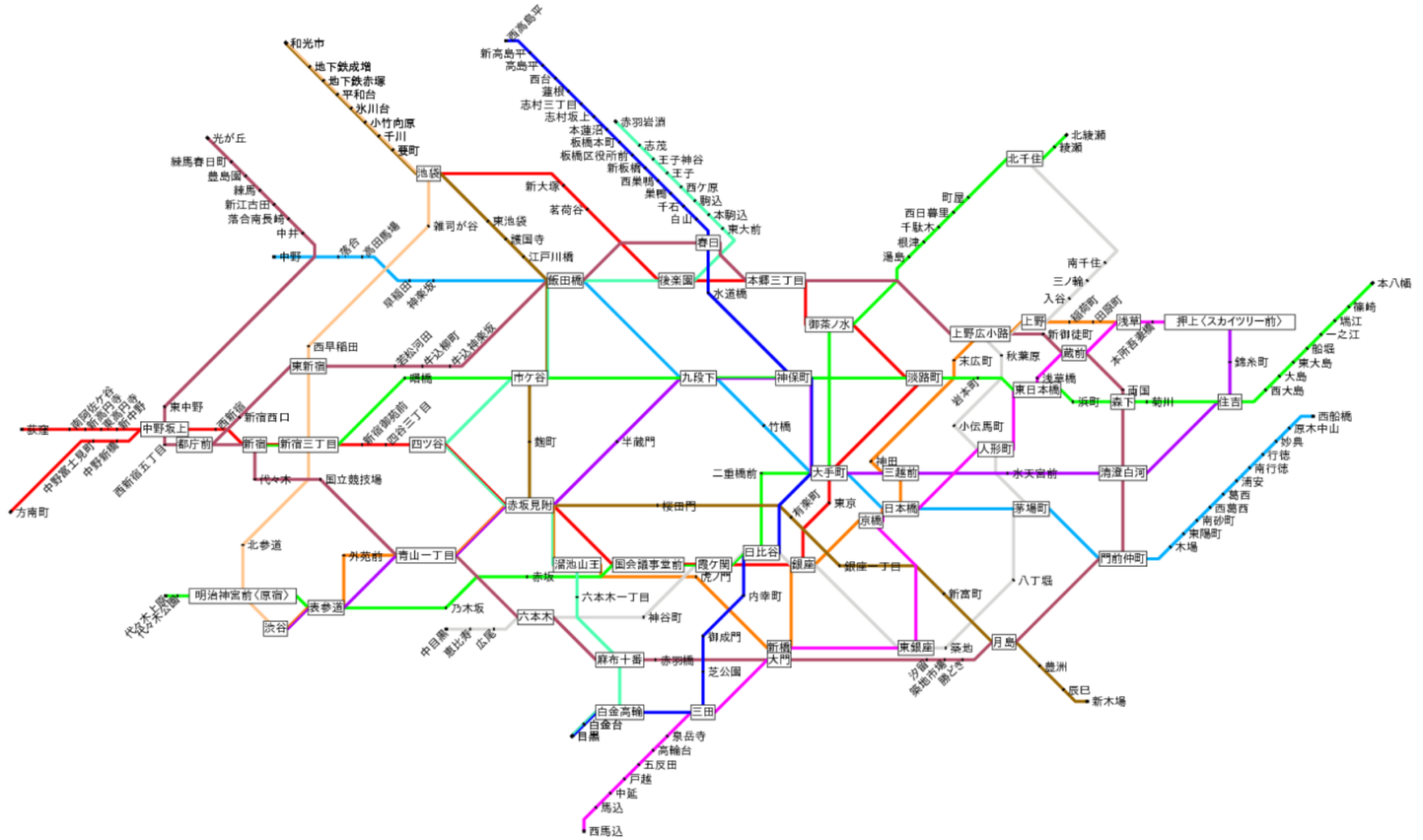
Example: Sydney & Tokyo



official map



Tokyo



- multi-phase MIP model [Onda, Moriguchi, Imai 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

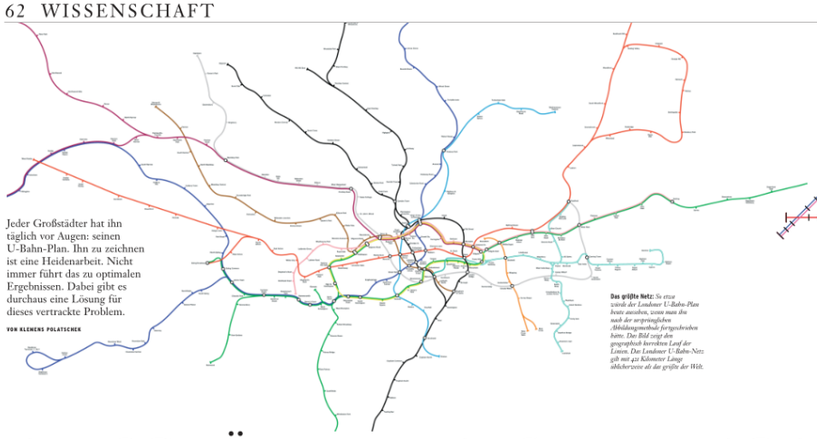
Pros.

- 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.

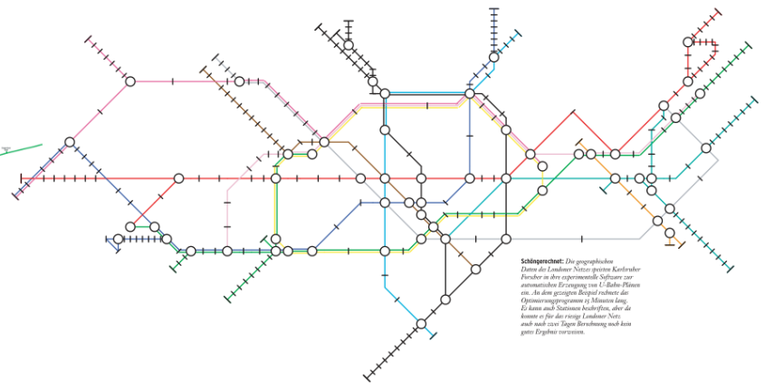
- long, sometimes unpredictable running times
- for large labeled networks no proof of optimality
- solutions only as good as the model specification

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 etwa 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 allerdersetzt als das größte der Welt.*



Schöngerechnet: *Die geographischen Daten des Londoner Netzes speisten 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 bearbeiten, 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 Stoten des Ungelösten auszugsuchen. Schon Wolffs Diplomarbeit an der Freien Universität Berlin hatte ihn 1992 mit der optimalen Beschriftung von Lageplänen 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 Blestifte, 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 weltmaßstrecendes Problem. Aber eines zum Verbiegen. Und es ist ein Problem für Handgreiflichkeiten, denn für den geschulten Informationsdesigner ist die Detailfindfindung ungedehnte Netzschemata ein Debattengenerator 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 Ruby 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 zus 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 hefrig 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 Kladderadatsch 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, per Handy und Flex Wip zu berechnen wären.

Als Wolff und Nöllenburg ihren himorischen Henry Beck zu Ende studiert und lange über einem Spiel moderner U-Bahn-Pläne gegrübelt hatten, erdachten sie siebenschlichte 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, das 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 vorbeiführenden 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 gerägt. 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 um 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 versucht auf einmal, wovon die zahllosen Skizzenzettel handelten, mit denen Henry Beck über Jahrzehnte hinweg sein Heim pflasterte. Das waren ebenfalls Optimierungsläufe, gekrönt vom 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.

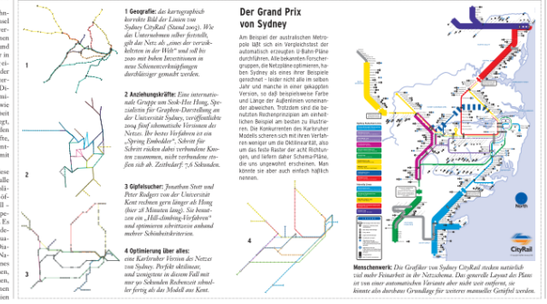
Man muß nur manchmal Geduld haben. Mathematisch gehört die U-Bahn-Aufgabe nämlich zu einer ebenso häufigen wie unbeliebten Klasse von Rechenproblemen, die man „NP-vollständig" nennt. Das ist der Beweis, den Martin Nöllenburg führte. Nach heutiger Erkenntnis gibt es bei Problemen dieser Klasse keinen 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 gerägt, desto 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 um 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 versucht auf einmal, wovon die zahllosen Skizzenzettel handelten, mit denen Henry Beck über Jahrzehnte hinweg sein Heim pflasterte. Das waren ebenfalls Optimierungsläufe, gekrönt vom 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.



1 Geografie: *Als kartographisch korrekte Bild der Linien von Sydney CityRail (Stand 2003). Wie das Unternehmen selber freiwillig gibt das Netz als „eines der verzicktbsten in der Welt" und soll bis 2010 mit hohen Investitionen in neue Schienenverbindungen durchlässiger gemacht werden.*

2 Anziehungskräfte: *Eine internationale Gruppe um Seok-Hee Hong, Spezialistin für Graphen-Darstellung an der Universität Sydney, veröffentlichte 2004 fünf schematische Versionen des Netzes. Die beste Verfahren ist ein „Spring Embedder". Schritt für Schritt rucken dabei verbundene Knoten zusammen, nicht verbundene stoßen sich ab. Zeitbedarf: 7,8 Sekunden.*

3 Gipfelsucher: *Jonathan Stott und Peter Rodgers von der Universität Kent rechnen gern länger als Hong (hier 28 Minuten lang). Sie benutzen ein „Hill-climbing-Verfahren" und optimieren schrittweise anhand mehrer Schönheitskriterien.*

4 Optimierung über alles: *eine Karlsruher Version des Netzes von Sydney. Perfekt aktzlinear, und sowogleiner in diesem Fall mit nur 90 Sekunden Rechenzeit schnellter fertig als das Modell aus Kent.*

Der Grand Prix von Sydney

Am Beispiel der australischen Metropole läßt sich ein Vergleichstest der automatisch erzeugten U-Bahn-Pläne durchführen. Alle bekannten Forschergruppen, die Netzpläne optimieren, haben Sydney als eines ihrer Beispiele genommen – leider nicht alle im selben Jahr und manche in einer gekappten Version, so daß beispielsweise Farbe und Länge der Aufreinen voneinander abweichen. Trotzdem sind die benutzten Rechenprinzipien am einheitlichen Beispiel am besten zu illustrieren. Die Konkurrenten des Karlsruher Modells schreiben sich mit ihren Verfahren weniger um die Oktilinearität, also um das feste Raster der acht Richtungen, und liefern daher Schema-Pläne, die uns ungewohnt erscheinen. Man könnte sie aber auch einfach häßlich nennen.

Menschenwerk: *Das Grafiker von Sydney CityRail stehen natürlich auf mehr Freiarbeit in der Netzschema. Das gewerbe Layout des Plans ist von einer automatischen Variante aber nicht weit entfernt, sie könnte als durchaus Grundlage für weitere manuelle Getüftel werden.*



Das Original: *Henry Becks erster Fahrplan 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 wurde eine Methode Tradition und ein weiterer Verhält geschaffen. In Berlin vergaß man die Vorzüge des Schemas bald wieder und kehrte lange zur geographischen Karte zurück. Man hatte wohl mit anderen Unwirrsen zu tun.*



Verwirrung in Manhattan: *Die Neu Yorker U-Bahn gilt nach London als zweitgrößte, es hat sogar mehr Stationen, 1972 fährte in diesen neuen „Subway Guide" nach dem Muster von Beck ein, gestaltet vom Star-Designer Massimo Vignelli. Der Plan wurde wieder abgeschafft. Die Neu 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 auch seine Dienstprobleme: in London. Man schrieb 1908, da waren die damals privaten Londoner U-Bahnen in Not. Bire sinsdreure Maulwurfarbeit 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.

Bild 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 ließen also in Einzelheiten weg, sie fingen sogar widerwillig an, der Karte einen gleitenden Maßstab zu geben, im Zentrum größer als außen. Trotzdem erinnerte der Plan schließlich nur noch an verwickelte Spaghetti beschungsweise 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 Geistesblitzlitz. Warum nicht die Geographie beiseite lassen! Die scherte den Untergrundspendler ja gar nicht. Alle Linien geradezeichen und aufräumen, sie horizontal und vertikal verteilen wie auf einem elektrischen Schaltbild, 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 Fahplan wurde im Januar 1933 gedruckt und vom Publikum ins Herz geschlossen. Die erste Auflage von 500 000 waren rasch vergriffen, im Februar mußten schon 100 000 nachgedruckt werden. Im Sommer 1933 stellte die gesamte gerade integrierte U-Bahn ihre Plakate auf das Schema Beck um.

1994 verfaßte der einflußreiche britische Designer Ken Garland das Standardwerk über Becks Arbeiten. Er wollte der Genehmigung 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 perfekten Arbeitsvertrages 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 Rauten, dann mußte er Stationsnamen in vorgrößerte Rauten schreiben oder Punkte in die Kreise setzen, wo auch ein Fettbalzbalzel 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 Irrsinn nachwies.

Ken Garland vermutet, daß sich Beck dem beugte, um bloß seine Arbeit 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 dipierte ihn deren Chef Harold F. Hutchinson und veröffentlichte mit Fanfarenstoß einen komplett neu gestalteten Plan, den er selbst angefertigt hatte. Er war bei näherer Betrachtung eine anmurmurhafte Karikatur der Beckschen Ideen. Nach zwei quälenden Jahren wurde das neue Schema nach Beckschen Prinzipien geheilt und bis heute fortgeschrieben.

Beck selbst 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, Ankünden 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 2011]

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 2011]

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 a focus route, but generalizes to entire maps

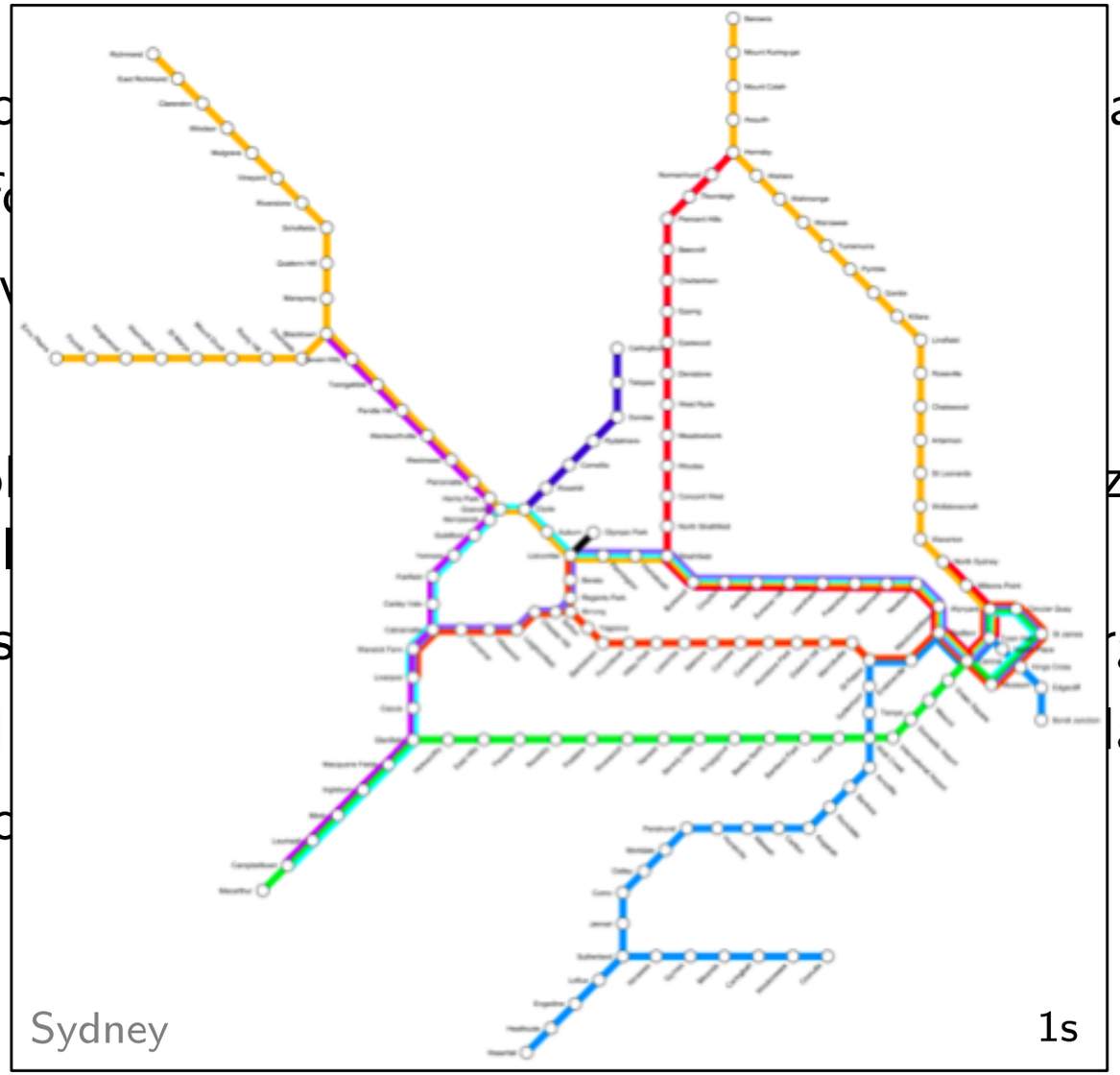
Least-Squares Schematization

[Wang & Chi 2011]

Idea.

- model layo … ared) energy terms
- variables fo …
- use iterativ …

- 3-step approach
  - compute topol … zing angular resolution **(R5)**, uniform edge l …
  - octilinearly dis … rgy term
  - optimize label … layout
- described for a fo …



Least-Squares Schematization

[Wang & Chi 2011]

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 a focus route, but 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

Summary

- variety of layout methods evolved over the last 15–20 years
- many shared design rules

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

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

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
- take thick edge bundles and large vertices into account

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, Gemsa, Nöllenburg, Pajor '10] Path Schematization for Route Sketches
- [Brandes & Pampel '09] On the Hardness of Orthogonal-Order Preserv-ing Graph Drawing
- [Gemsa, Nöllenburg, Pajor, Rutter '11] On d-Regular Schematization of Embedded Paths
- [Hong, Merrick, do Nascimento '06] Automatic visualisation 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, Taylor, Anand, Thomas '06] Automated production of schematic maps for mobile application
- [Ware & Richards '13] An ant colony system algorithm for automatically schematizing transport network data set
- [Stott, Rodgers, Burkhard, Meier, Smis '11] Automatic 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