Efficient Generation of Geographically Accurate Transit Maps

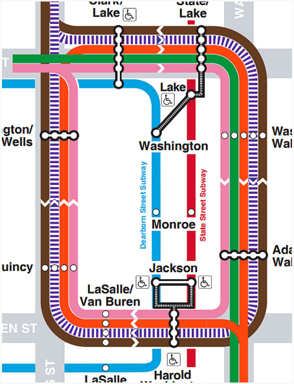
Hannah Bast[1], Patrick Brosi[1] and Sabine Storandt[2]

[1] University of Freiburg
[2] LMU Würzburg

26th ACM SIGSPATIAL - Seattle, Washington, USA

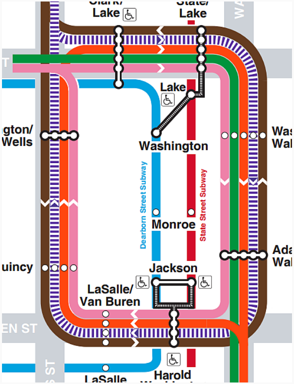
Motivation

Official CTA map

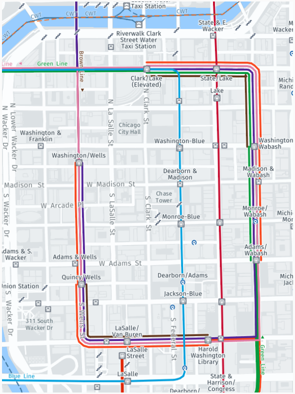


Motivation

Official CTA map

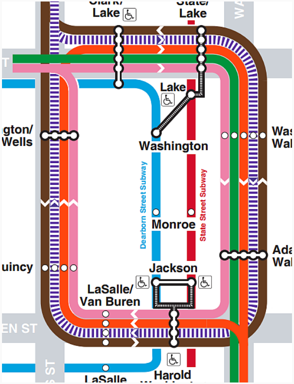


HERE

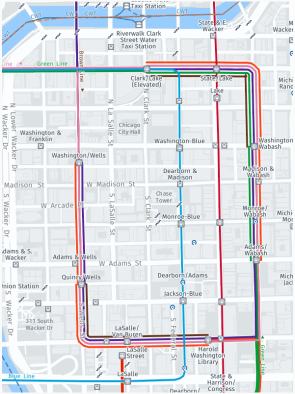


Motivation

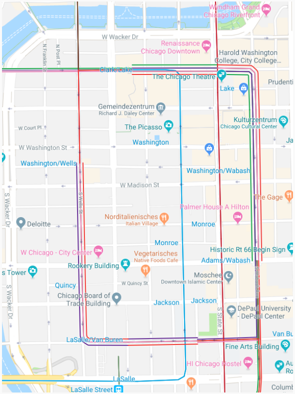
Official CTA map



HERE



Google



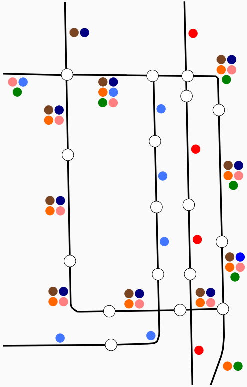
Goal

Goal: Generate these maps automatically, in high quality

”Bag of trips”

Goal

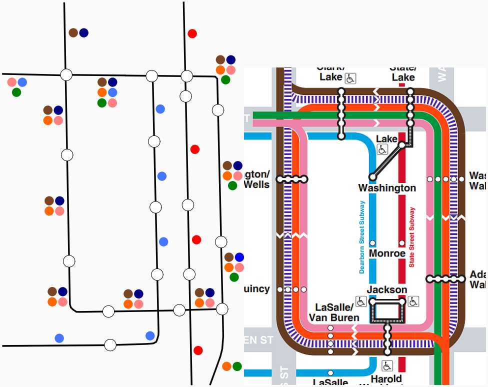
Goal: Generate these maps automatically, in high quality



”Bag of trips”

Goal

Goal: Generate these maps automatically, in high quality

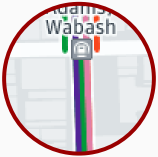


"Bag of trips"

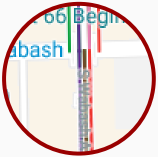
Challenges

Official

HERE



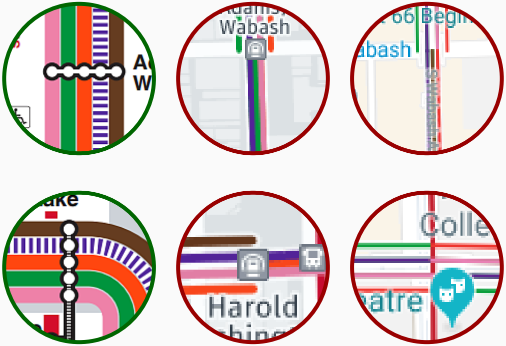
Google



I. Avoid line overlaps

Challenges

Official HERE Google

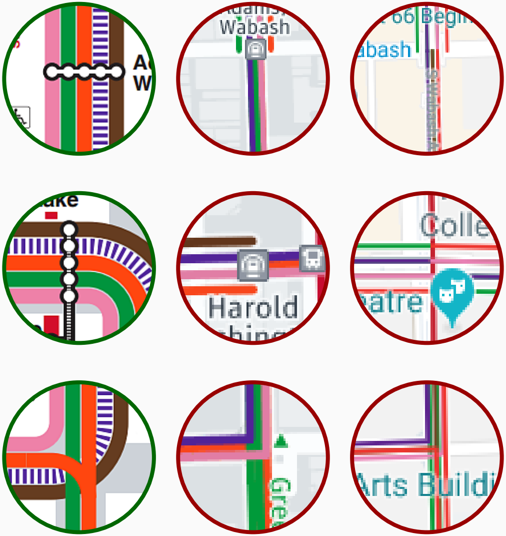


I. Avoid line overlaps

II. Match line orderings

Challenges

Official HERE Google



I. Avoid line overlaps

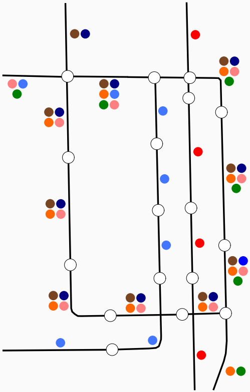
II. Match line orderings

III. Clearly indicate line continuations

Line graph construction

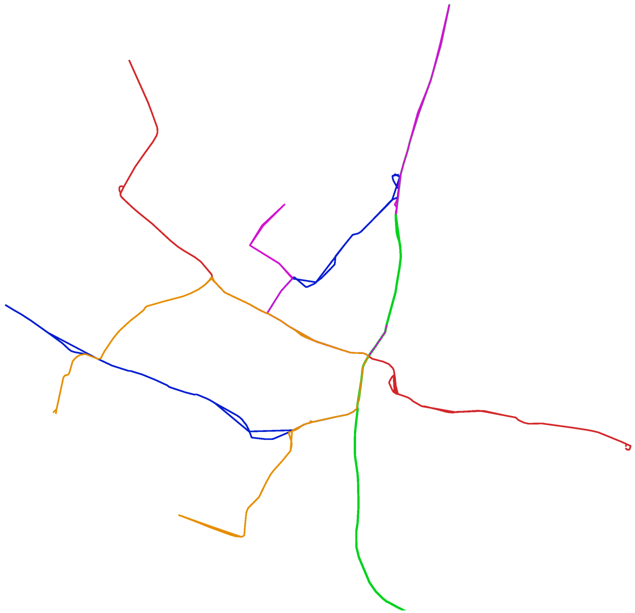
Line graph:

- Undirected labeled graph $G = (V, E, L)$
- Edge labels are subsets of the network lines $\mathcal{L}$ ($L(e) \subseteq \mathcal{L}$)
- Nodes are usually stations

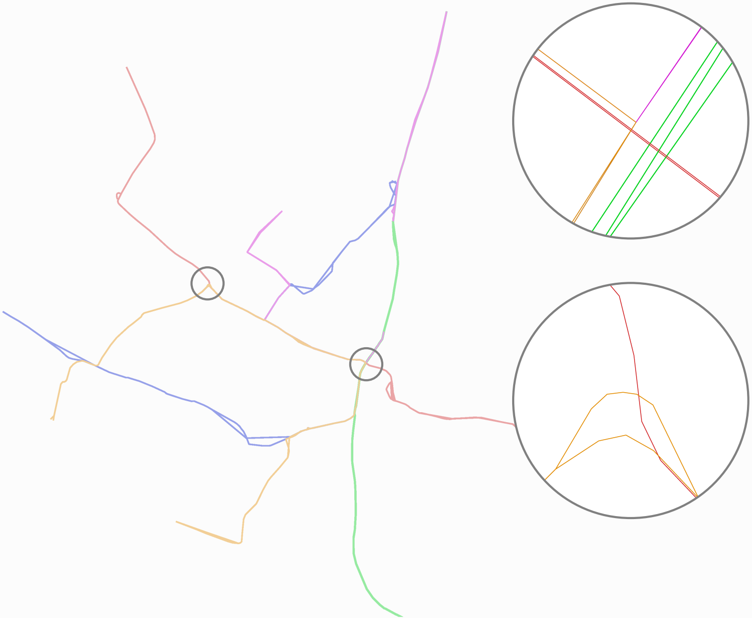


Example: $\mathcal{L} = \{\}$ and $L((a,b)) = \{\}$

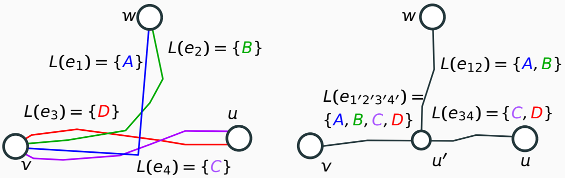
Line graph construction - Input data



Line graph construction - Input data

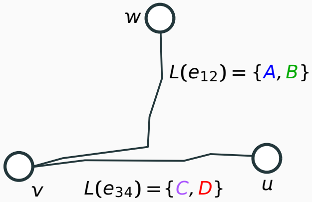


Line graph construction - Shared segment collapsing

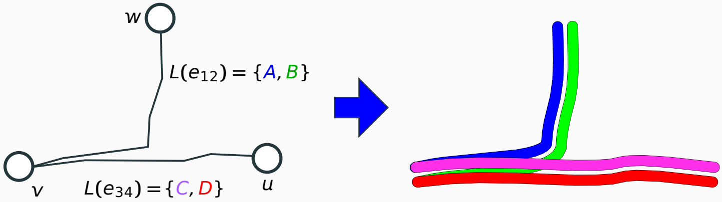


- Repeatedly collapse (segments of) two edges e and f within a distance $\hat{d}$
- Sweep over some edge e in steps of 10 m, measure distance d of current point on e to f
- If $d < \hat{d}$, start new segment. If not, end current (if open)
- Take average between the two ”shared segments” on e and f
- Add additional non-station nodes at segment boundaries

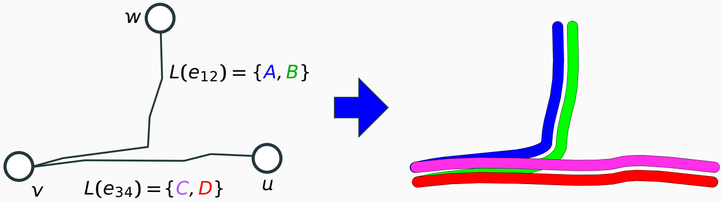
Line graph construction - Why non-station nodes?

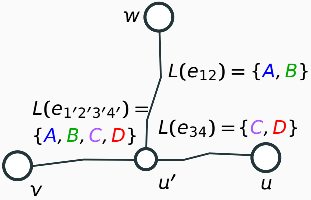


Line graph construction - Why non-station nodes?

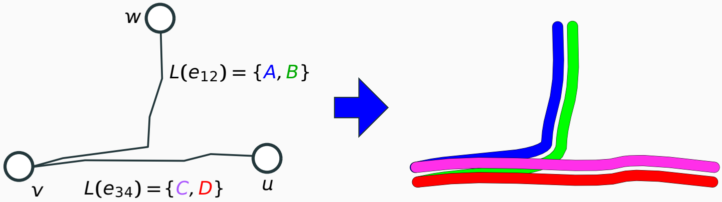


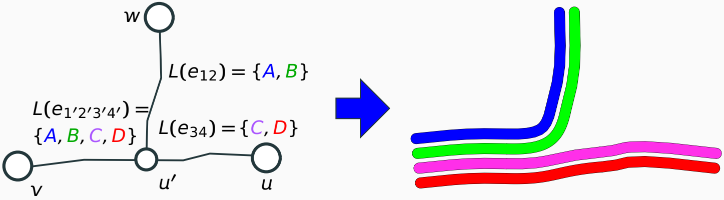
Line graph construction - Why non-station nodes?



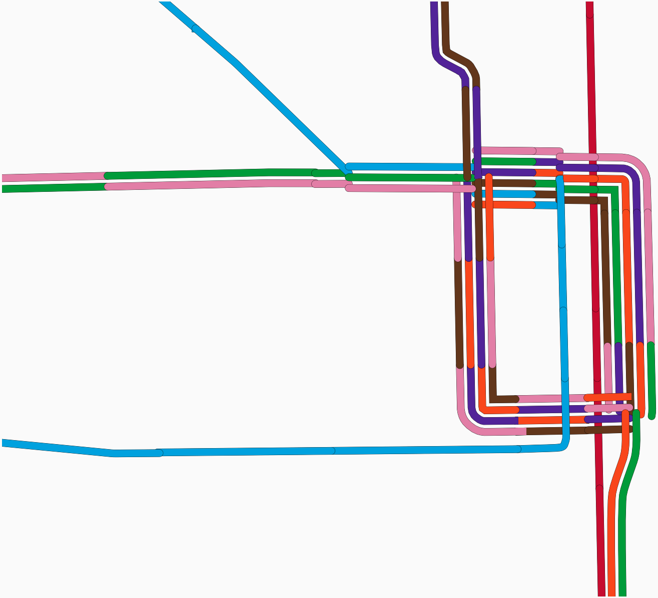


Line graph construction - Why non-station nodes?

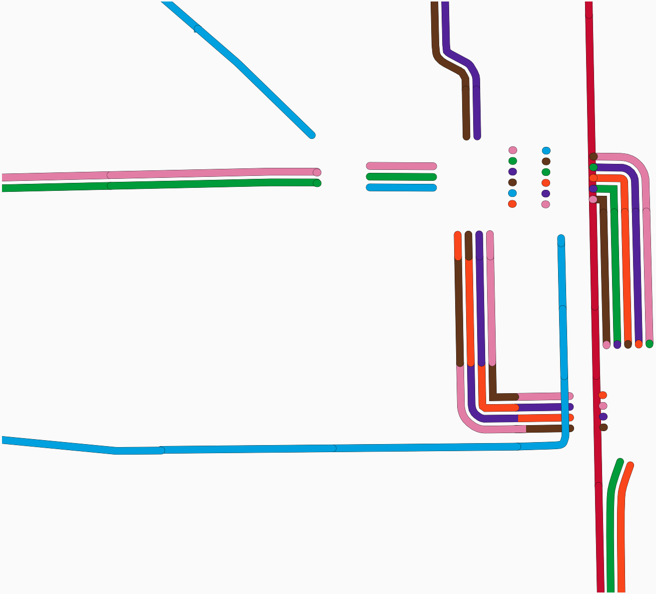




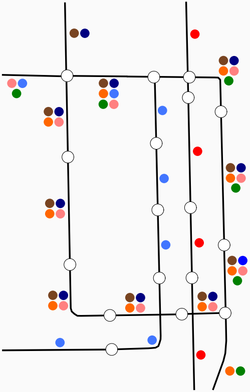
Results so far (1)



Results so far (1)



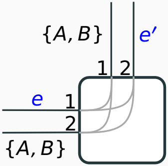
Line-ordering optimization

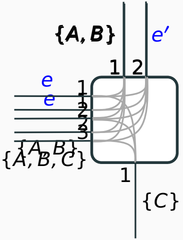


- 23 edges
- Each edge e has $|L(e)|!$ possible line permutations
- Possible configurations for the graph on the left: $> 2 \times 10^{17}$

$\Rightarrow$ **Naive exhaustive search infeasible**

Line-ordering optimization - Baseline ILP





- For each edge e, line l and position p, introduce variable $x_{elp} \in 0, 1$
- Example: x_{eA1} and x_{eA2} for line A
- Constraint: all x_{elp} have to sum up to 1 for a single line l on a single edge e
- Standard crossing: Objective variable $x_{ee'AB}$ which is 1 if $p_e(A) < p_e(B)$ and $p_{e'}(A) > p_{e'}(B)$, or else 0
- Split crossing: Objective variable $x_{ee'e''AB}$ which is 1 if $p_e(A) < p_e(B)$, or else 0

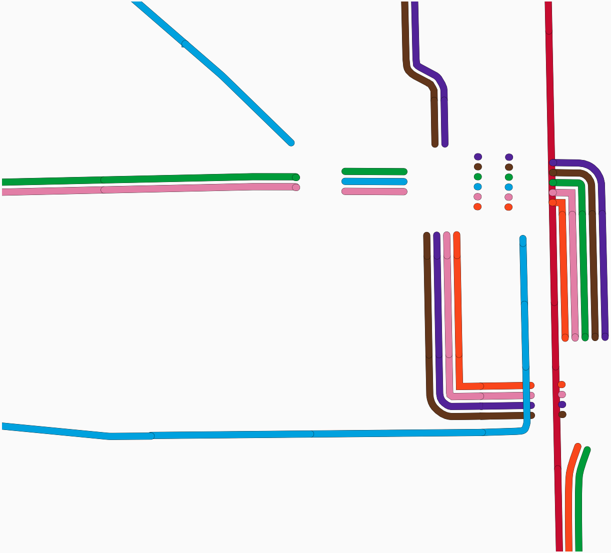
$\Rightarrow \mathcal{O}(|E|M^2)$ variables, $\mathcal{O}(|E|M^6)$ constraints

Line-ordering optimization - Improved ILP

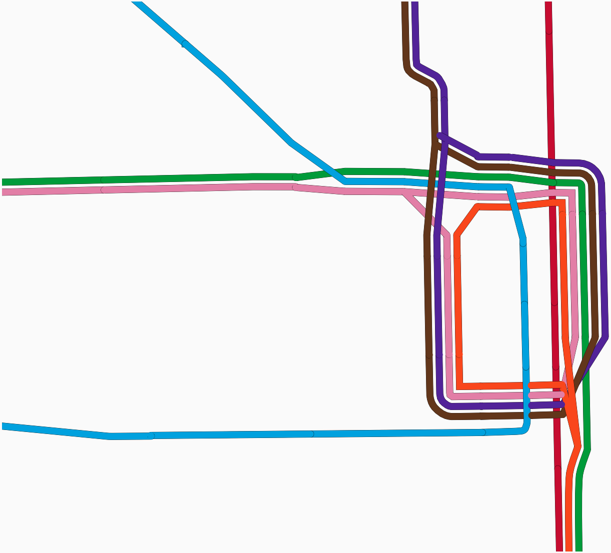
- **Observation:** we only need to check if $p_e(A) < p_e(B)$ (or vice versa) for both types of crossings
- But we explicitly enumerate all possible line positions of A and B on e
- Basic idea: introduce binary variables $x_{eA<B}$ and $x_{eB<A}$ which can be efficiently checked

$\Rightarrow \mathcal{O}(|E|M^2)$ variables, $\mathcal{O}(|E|M^2)$ constraints

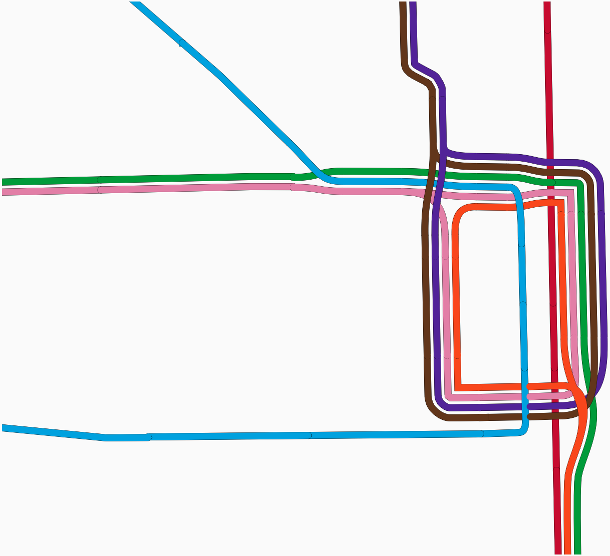
Results so far (2)



Results so far (2)



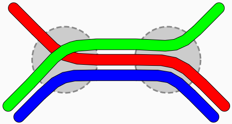
Results so far (2)



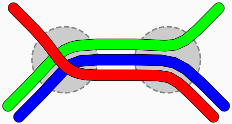
Line-ordering optimization - Line separations

vs

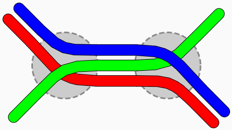
vs



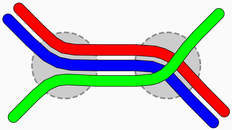
1 crossing, 1 separation



2 crossings, 0 separations

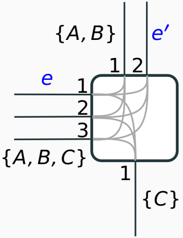


2 crossings, 1 separation



2 crossings, 0 separations

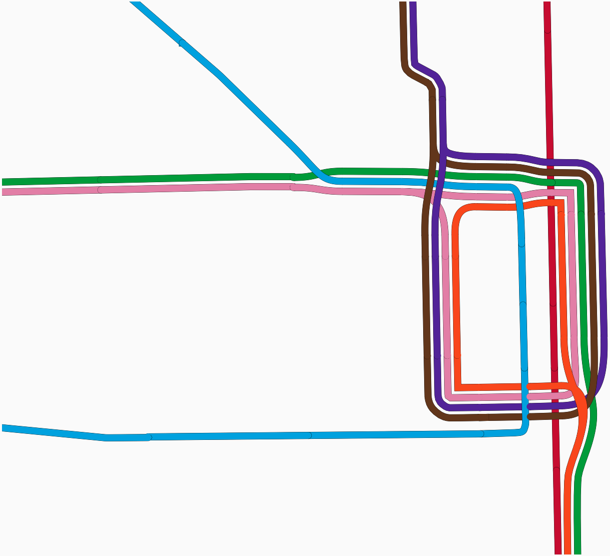
Line-ordering optimization - Line separations (ctd.)



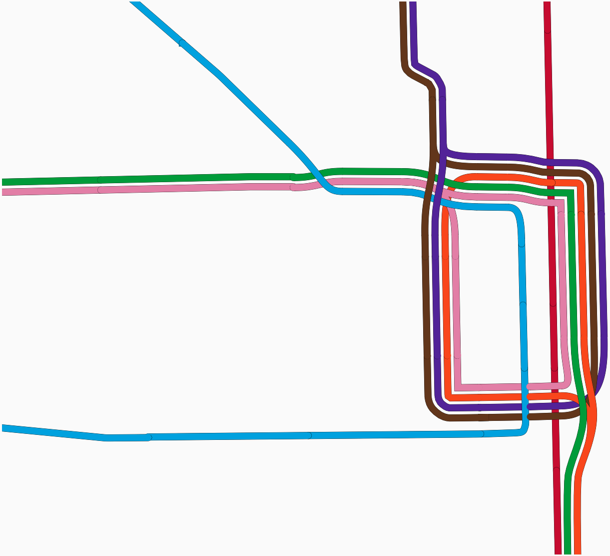
- **Idea:** If two lines A, B continue from e to e', set a binary splitting variable $x_{ee'A\|B} = 1$ if they are next to each other in e, but no in e'
- Add $x_{ee'A\|B}$ to the objective function

$\Rightarrow$ Still $\mathcal{O}(|E|M^2)$ variables, $\mathcal{O}(|E|M^2)$ constraints

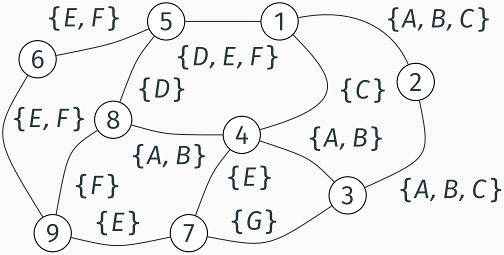
Results so far (3)



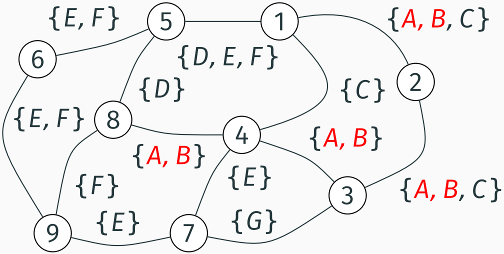
Results so far (3)



Line-ordering optimization - Core optim graph

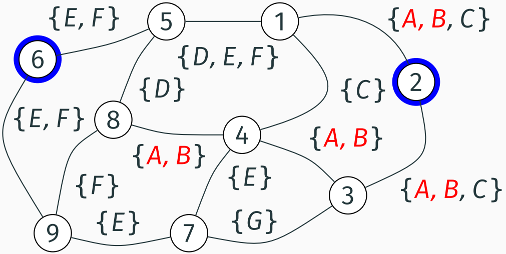


Line-ordering optimization - Core optim graph



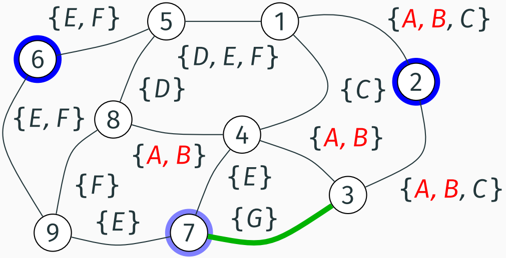
- Combine lines *A*, *B* that always occur together into a single new line ($A, B \longrightarrow X$)

Line-ordering optimization - Core optim graph



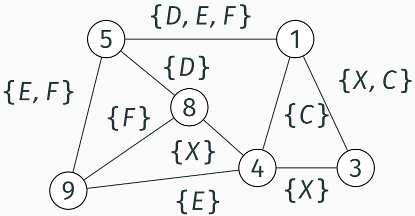
- Combine lines A, B that always occur together into a single new line ($A, B \longrightarrow X$)
- Delete nodes with degree 2 and merge their adjacent edges

Line-ordering optimization - Core optim graph



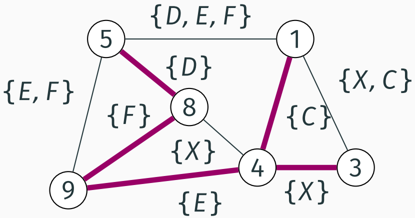
- Combine lines *A*, *B* that always occur together into a single new line ($A, B \rightarrow X$)
- Delete nodes with degree 2 and merge their adjacent edges
- Remove edges (u, v) where u and v are termini for all $L((u, v))$

Line-ordering optimization - Core optim graph



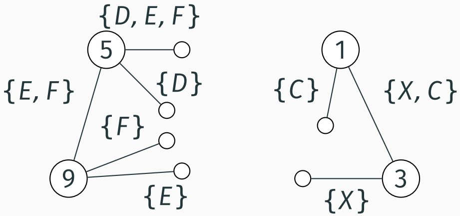
- Combine lines A, B that always occur together into a single new line ($A, B \longrightarrow X$)
- Delete nodes with degree 2 and merge their adjacent edges
- Remove edges (u, v) where u and v are termini for all $L((u, v))$

Line-ordering optimization - Core optim graph



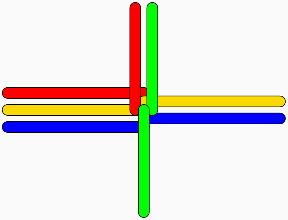
- Combine lines A, B that always occur together into a single new line ($A, B \to X$)
- Delete nodes with degree 2 and merge their adjacent edges
- Remove edges (u, v) where u and v are termini for all $L((u, v))$
- Cut edges with $|L(e)| = 1$

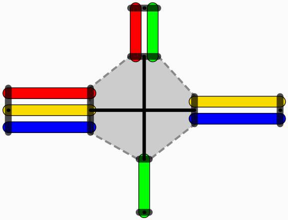
Line-ordering optimization - Core optim graph

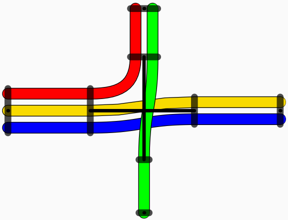


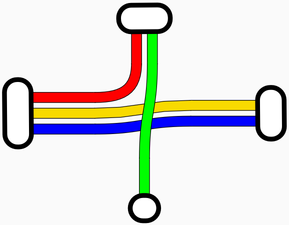
- Combine lines A, B that always occur together into a single new line ($A, B \longrightarrow X$)
- Delete nodes with degree 2 and merge their adjacent edges
- Remove edges (u, v) where u and v are termini for all $L((u, v))$
- Cut edges with $|L(e)| = 1$

Rendering

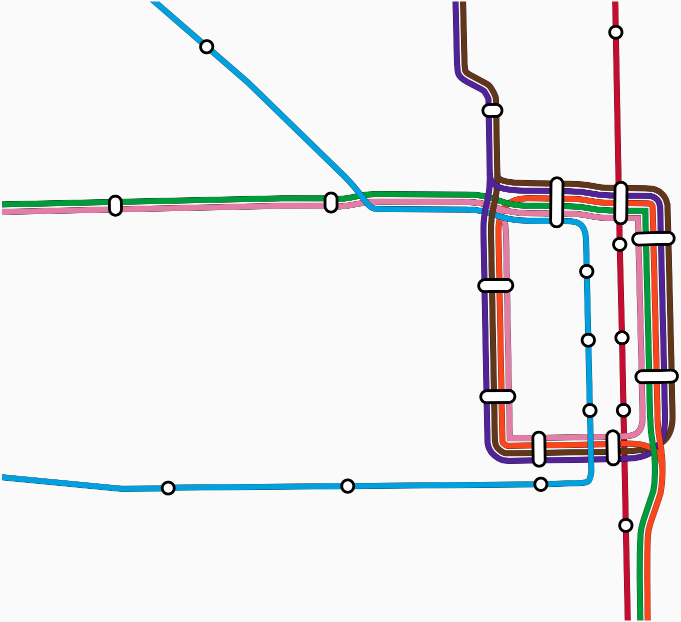








Results so far (4)



Evaluation

Future work

- Additional rules for core graph reduction (work in progress)
- Faster construction times of line graph
- TODO

Thank you!