Inverse shortest paths in directed acyclic graphs

Presentation for KLAIM 2023

Sven O. Krumke, Florian Dahms, Orges Leka

Technische Hochschule Bingen

o.leka@th-bingen.de

Presentation

September 22, 2023

Overview

Problem definition of Burton und Toint 1992

- Given: A directed graph $G = (V, E)$, costs c on the edges, a set P of paths.
- we are searching for: costs $\hat{c}$ on the edges, such that:
- $\hat{c} := \operatorname{argmin}_{\hat{c}} \sum_{(u,v)\in E} (c(u,v) - \hat{c}(u,v))^2$
- under constraint: $\forall p \in P$ ist p is a shortest path under $\hat{c}$

possible Application:

- given: A directed graph $G = (V, E)$, costs c on the edges, a set P of paths.
- vertices correspond here to railway routes so called 'Fahrwege'
- edges correspond to connections of Fahrwege
- Set of paths $p \in P$ corresponds to paths from given schedules which were constructed manually

Heuristic Idea

- Divide the problem of Burton and Toint into two tasks:
- Desired: Costs $\hat{c}$ on the edges such that:
- 1. For all $p \in P$, p is a shortest path under $\hat{c}$
- 2. Modify $\hat{c}$ to $c^*(u,v) := \hat{c}(u,v) + h_v - h_u$, where:
- $c^* := \text{argmin}_h \sum_{(u,v)\in E}(c(u,v) - (\hat{c}(u,v) + h_v - h_u))^2$

1st Step in the Heuristic Idea: Straight Line as a Shortest Line

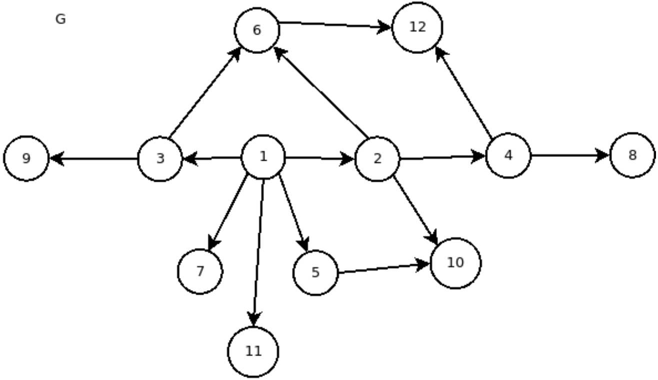


Figure: Example graph G

1st Step in the Heuristic Idea: Straight Line as a Shortest Line

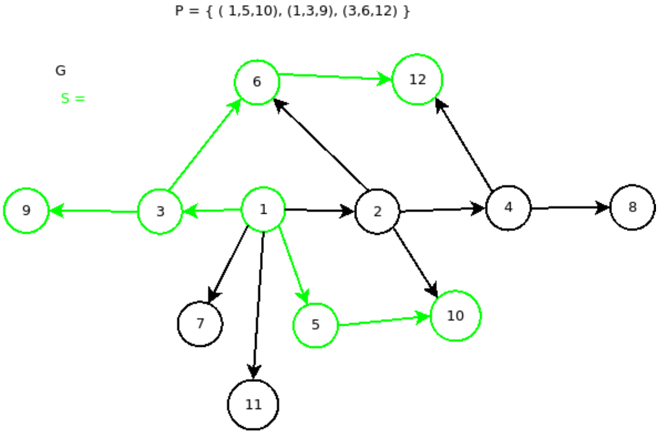
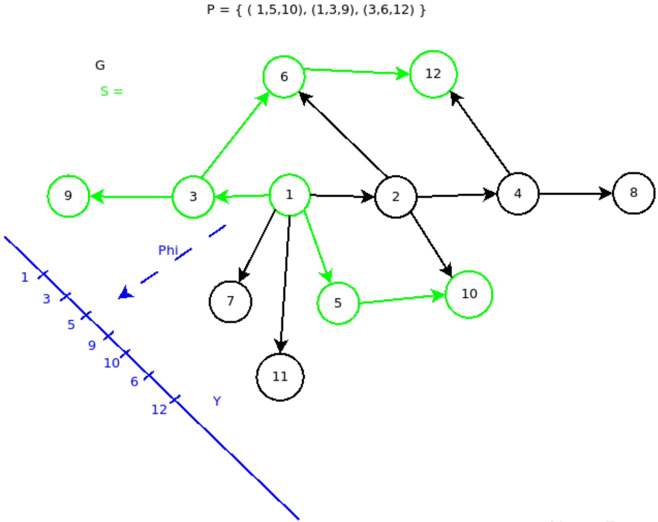


Figure: Preserve prescribed paths in subgraph S

1st Step in the Heuristic Idea: Straight Line as a Shortest Line



1st Step in the Heuristic Idea: Straight Line as a Shortest Line

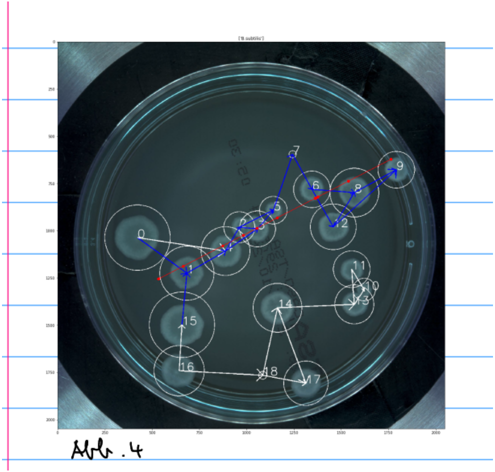


Figure: blue=Prescribed paths, red=Projection line, white=Graph

2nd Step in the Heuristic Idea: Use Potential Function h_v to Modify Costs

Given:

- $\mathbf{c}(u, v) :=$ Currently used costs
- $\hat{c}(u, v) :=$ Shortest path costs calculated in the 1st step

2nd Step in the Heuristic Idea: Use Potential Function h_v to Modify Costs

Desired: Potential function $h : V \to \mathbf{R}$ that minimizes $(*)$:

$$(*) : \min_h \sum_{(u,v)\in E} (\mathbf{c}(u,v) - (\hat{c}(u,v) + h_v - h_u))^2$$

$$\text{and } \gamma(u,v) := \hat{c}(u,v) + h_v - h_u \geq^! 0$$

2nd Step in the Heuristic Idea: Use Potential Function h_v to Modify Costs

Two Ideas:
(i) Formulate (*) as an optimization problem and solve it with an optimizer
(ii) Use a heuristic to solve (*) quickly (Idea from Johnson's or Surballee's Algorithm)

Extension using Kernel Trick

Observation:
The heuristic method only uses distances and inner products in the calculations
This inner product can be replaced by a positive definite kernel k
Distance: $d(x, y) = \sqrt{k(x, x) + k(y, y) - 2k(x, y)}$

Extension using Kernel Trick

Advantage:
Properties of routes such as:

- Number of switches
- Number of intersections
- Length
- Passenger exit yes/no
- Maximum speed

can be taken into account in the cost calculation of the distance function.
Different distance functions could be defined and empirically tested for suitability.

Graph G with Distance Costs

$$c(u,v) := \sqrt{|\min(u,u) + \min(v,v) - 2 \cdot \min(u,v)|} = \sqrt{|u-v|}$$

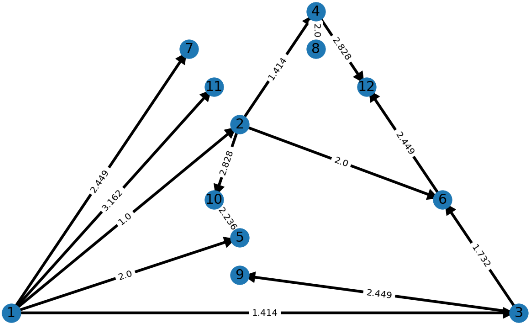


Figure: Graph G with Distance Costs

Graph H with Currently Used Costs

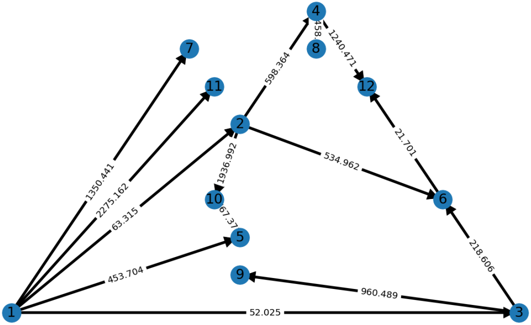


Figure: Graph H with Currently Used Costs

Graph I with Costs for Shortest Paths

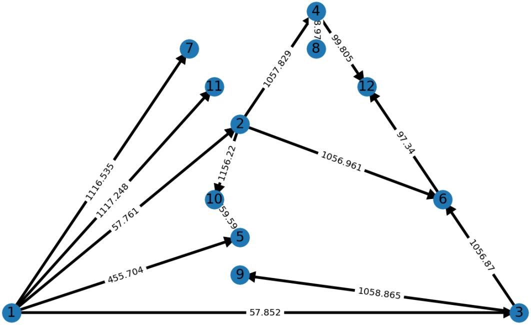


Figure: Graph I with Costs for Shortest Paths

Prescribed Paths

paths = [(1, 2, 6, 12), (1, 3, 6, 12)]
shortest path as required: (1, 2, 6, 12) 1212.06 1212.06
shortest path as required: (1, 3, 6, 12) 1212.06 1212.06
not a shortest path as required: (1, 2, 4, 12) 1215.39 1212.06

Runtimes for the 1st Step in the Heuristic

	nrNodesInGraph	nrEdgesInGraph	nrRndPaths	graphCreationTimeInSec	rndPathCreationTimesInSec	graphProjectionTimesInSec
0	100	171	5	0.008509397506713867	0.008020401000976562	0.15915966033935547
1	100	171	10	0.008509397506713867	0.0007266998291015625	0.017099857330322266
2	100	171	15	0.008509397506713867	0.0009381771087646484	0.01642012596130371
3	500	1008	25	0.05508828163146973	0.00228118896484375	0.09429073333740234
4	500	1008	50	0.05508828163146973	0.004318714141845703	0.09512495994567871
5	500	1008	75	0.05508828163146973	0.00675201416015625	0.09413337707519531
6	1000	2126	50	0.12341570854187012	0.007304668426513672	0.2034435272216797
7	1000	2126	100	0.12341570854187012	0.013124465942382812	0.22152495384216309
8	1000	2126	150	0.12341570854187012	0.021610498428344727	0.20542025566101074
9	5000	11738	250	0.86210036277771	0.11178779602050781	1.164170742034912
10	5000	11738	500	0.86210036277771	0.22905516624450684	1.1609125137329102
11	5000	11738	750	0.86210036277771	0.35599493980407715	1.1684775352478027
12	10000	24300	500	1.7901675701141357	0.4459238052368164	2.4443984031677246
13	10000	24300	1000	1.7901675701141357	0.8848965167999268	2.485508918762207
14	10000	24300	1500	1.7901675701141357	1.3129947185516357	2.4497201442718506
15	100000	266400	5000	30.827675580978394	51.31938123703003	32.245964765548706
16	100000	266400	10000	30.827675580978394	104.8054690361023	31.861108779907227
17	100000	266400	15000	30.827675580978394	154.85676980018616	32.35290741920471
18	1000000	2853708	50000	411.71499848365784	5268.441532611847	478.899352312088
19	1000000	2853708	100000	411.71499848365784	10475.782470464706	479.71676087379456
20	1000000	2853708	150000	411.71499848365784	16124.274740934372	470.5861692428589

Figure: Result: Heuristic 1st Step, Runtimes in seconds

Thank you for your attention

Any Questions? Comments?

Code and paper may be found here:
https://github.com/githubuser1983/klaim2023