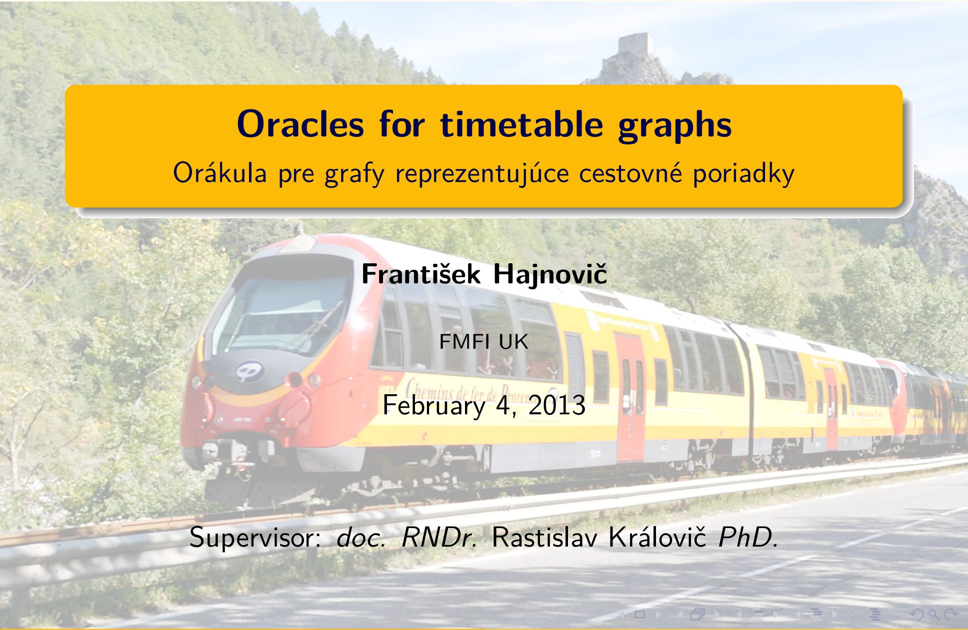


Oracles for timetable graphs

Orákula pre grafy reprezentujúce cestovné poriadky

František Hajnovič

FMFI UK

February 4, 2013

Supervisor: *doc. RNDr.* Rastislav Královič *PhD.*

Content

Introduction

Introduction

What is it about?

- **Earliest arrival problem** (EAP) given a timetable
  - EA only
  - Connection also

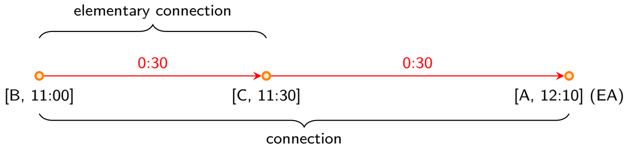
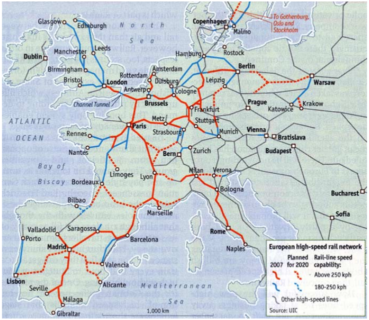


Figure : **Connection**, **elementary connection** and **earliest arrival**

Motivation & usage

- Timetable search engines (*cp.sk*, *imhd.sk*...)
- Bigger scale (e.g. Europe-wide)



Timetable and underlying graph

Place		Time	
From	**To**	**Departure**	**Arrival**
A	B	10:00	10:45
B	C	11:00	11:30
B	C	11:30	12:10
B	A	11:20	12:30
C	A	11:45	12:15

Table : **Timetable** - a set of **elementary connections** (between pairs of **cities**)

Timetable and underlying graph

Place		Time	
From	To	Departure	Arrival
A	B	10:00	10:45
B	C	11:00	11:30
B	C	11:30	12:10
B	A	11:20	12:30
C	A	11:45	12:15

Table : **Timetable** - a set of **elementary connections** (between pairs of **cities**)

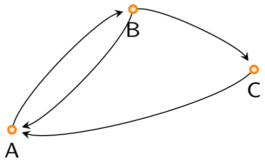


Figure : **Underlying graph**

Oracle

- Dijkstra's algorithm $\mathcal{O}(m + n \log n)$

Oracle

- Dijkstra's algorithm $\mathcal{O}(m + n \log n)$
- Precompute information $\rightarrow$ **Oracle based method**
  - *Preprocessing time*
  - *Size*
  - *Query time*
  - *Stretch*

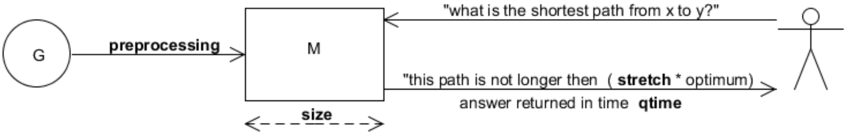


Figure : Oracle based method

Goals

- Devise methods to tackle EAP
- Analyse properties of timetables

Goals

- Devise methods to tackle EAP
- Analyse properties of timetables

Odchod	Príchod	Dĺžka cesty*	Použité linky	Zóny	Cena*
21:59	22:10	11 min	95	-	0,70 €
22:09	22:20	11 min	95	-	0,70 €
22:19	22:30	11 min	95	-	0,70 €
22:29	22:40	11 min	95	-	0,70 €
22:39	22:50	11 min	95	-	0,70 €

Figure : Exploit redundancy in timetables?

Data

Data

Data

Name	Description	El. conns.	Cities	Time range	Height (h)
air01	domestic flights (US)	592767	250	1 month	24374
cpru	regional bus (SVK)	10011	250	1 day	239
cpza	regional bus (SVK)	15776	250	1 day	370
montr	public transport (Montreal)	7118	211	1 day	363
zsr	country-wide rails (SVK)	931647	233	1 year	59928

Table : Data - timetable properties

Underlying shortest paths

Underlying shortest paths

Idea

- *"Usually we go through the same sequence of cities"*

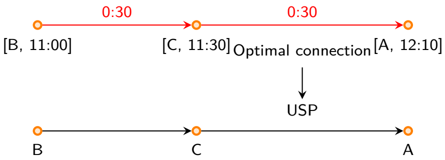


Figure : **Underlying shortest path**

Idea

- *"Usually we go through the same sequence of cities"*

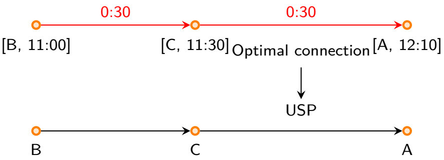


Figure : **Underlying shortest path**

Name	Overtaken edges (%)
air01	1%
cpru	2%
cpza	2%
montr	1%
zsr	0%

Table : Data - underlying graphs properties

- **Overtaking** causes problems, but can be easily removed

USP-OR

- Pre-compute all connections - space $\mathcal{O}(h\ n^3)$
    - height $h \gg n$

USP-OR

- Pre-compute all connections - space $\mathcal{O}(h\ n^3)$
  - height $h \gg n$
- Pre-compute all USPs - space $\mathcal{O}(\tau\ n^3)$
  - Exact answers, $\mathcal{O}(\tau\ n)$ query time
  - Preprocessing $\mathcal{O}((hn)^3)$
  - How big is τ?

USP-OR

- Pre-compute all connections - space $\mathcal{O}(h\ n^3)$
  - height $h \gg n$
- Pre-compute all USPs - space $\mathcal{O}(\tau\ n^3)$
  - Exact answers, $\mathcal{O}(\tau\ n)$ query time
  - Preprocessing $\mathcal{O}((hn)^3)$
  - How big is τ?

Name	avg $\tau_{A,B}$	max $\tau_{A,B}$
air01	18.3	126
cpru	10.25	53
cpza	5.87	45
montr	4.09	30
zsr	8.9	85

Table : $\tau_{A,B}$ - number of USPs between A and B

USP-OR

- Pre-compute all connections - space $\mathcal{O}(h\ n^3)$
  - height $h \gg n$
- Pre-compute all USPs - space $\mathcal{O}(\tau\ n^3)$
  - Exact answers, $\mathcal{O}(\tau\ n)$ query time
  - Preprocessing $\mathcal{O}((hn)^3)$
  - How big is τ?
  - Space too big anyway

Name	**avg** $\tau_{A,B}$	**max** $\tau_{A,B}$
air01	18.3	126
cpru	10.25	53
cpza	5.87	45
montr	4.09	30
zsr	8.9	85

Table : $\tau_{A,B}$ - number of USPs between A and B

USP-OR-A

- **Access nodes** - set A of cities in UG
  - Size $|Acc| = \mathcal{O}(\sqrt{n})$
  - Small node neighbourhoods $\forall v \ |neigh_{Acc}(v)| = \mathcal{O}(\sqrt{n})$
  - Few local access nodes $(\forall v \ |Acc_v| = \mathcal{O}(f(n)))$

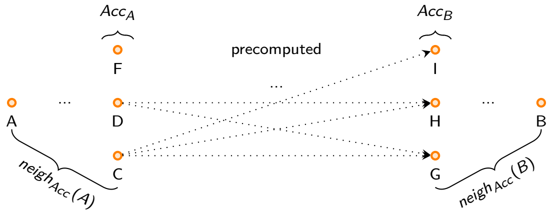


Figure : **Principle of access nodes**

USP-OR-A

- **Access nodes** - set A of cities in UG
  - Size $|Acc| = \mathcal{O}(\sqrt{n})$
  - Small node neighbourhoods $\forall v \; |neigh_{Acc}(v)| = \mathcal{O}(\sqrt{n})$
  - Few local access nodes $(\forall v \; |Acc_v| = \mathcal{O}(f(n)))$
- Inspiration by TRANSIT algorithm [BFM06]
  - 10 access nodes on average in road network

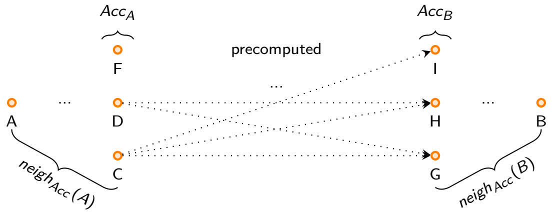


Figure : **Principle of access nodes**

USP-OR-A

- Space $\mathcal{O}(\tau\ n^2)$
- Query time $\mathcal{O}(\tau\ n\ f(n)^2)$
  - Search in neighbourhood can be Dijkstra

USP-OR-A

- Space $\mathcal{O}(\tau\ n^2)$
- Query time $\mathcal{O}(\tau\ n\ f(n)^2)$
  - Search in neighbourhood can be Dijkstra
- We may limit precomputed USPs

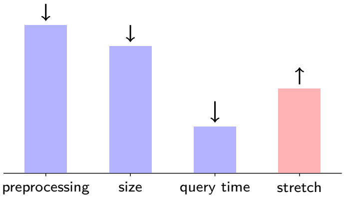


Figure : Decreasing τ to save resources

Existing methods

- **Time-dependent SHARC** [Del08], **Time-dependent CH** [BDSV09]
  - Speed-ups of about 26 / 1500, respectively
  - Meant for time-dependent routing in road networks
- **Time-expanded approach** [DPW09]
  - Speed-ups of about 56
  - Remodelling unimportant stations

Neural networks

Neural networks

Neural network approaches

- Multi-layer perceptron, back propagation
- Input layer = events + cities. Output layer:
  1. Arcs of UG *rightarrow* USP
  2. Arcs of UG *rightarrow* routing
  3. Earliest arrival value

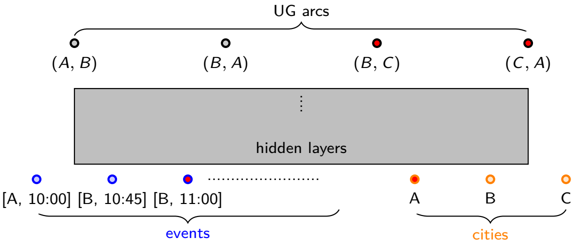


Figure : Approach 1.)

Neural network approaches

- Multi-layer perceptron, back propagation
- Input layer = events + cities. Output layer:
  1. Arcs of UG *rightarrow* USP
  2. Arcs of UG *rightarrow* routing
  3. Earliest arrival value
- Long training times
- Poor ability to find optimum ($< 50\%$)

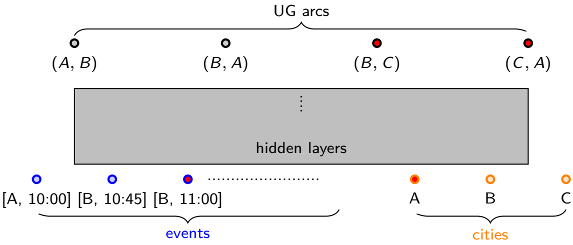


Figure : Approach 1.)

Application TTBlazer

Application TTBlazer

Timetable analyzer - TTBlazer

- Works with UG, TE, TD, TT
- Analysis (τ, HD, degrees...), oracles (USP-OR, Dijkstra...), modifications (remove overtaking...), generation (subgraphs, TT $\rightarrow$ TD ...)
- Running & evaluating tests
- Easily extendible

Figure : It's *blazing* fast!

Conclusion

Conclusion

Conclusion

- Trying out novel approaches to solving EAP in timetables
  - *USP-OR*: Exact and quick answers but high space and time preprocessing
  - *NN*: Problem too challenging for NN/try different types of network
- Analysis of **various** real-world timetables
  - Better insight on properties of timetables
- Useful and easily extendible application

To-do

- Find a good access node set
- Reduce the space complexity further
- Train and test properly neural network oracles

Bibliography I

[BDSV09] Gernot Veit Batz, Daniel Delling, Peter Sanders, and Christian Vetter. Time-dependent contraction hierarchies. In Irene Finocchi and John Hershberger, editors, *ALENEX*, pages 97–105. SIAM, 2009.

[BFM06] Holger Bast, Stefan Funke, and Domagoj Matijevic. Transit— ultrafast shortest-path queries with linear-time preprocessing, 2006.

[Del08] Daniel Delling. Time-dependent sharc-routing. In Dan Halperin and Kurt Mehlhorn, editors, *ESA*, volume 5193 of *Lecture Notes in Computer Science*, pages 332–343. Springer, 2008. ISBN 978-3-540-87743-1.

[DPW09] Daniel Delling, Thomas Pajor, and Dorothea Wagner. Engineering time-expanded graphs for faster timetable information. In Ravindra Ahuja, Rolf Möhring, and Christos Zaroliagis, editors, *Robust and Online Large-Scale Optimization*, volume 5868 of *Lecture Notes in Computer Science*, pages 182–206. Springer Berlin / Heidelberg, 2009. ISBN 978-3-642-05464-8.

Thank you for the attention