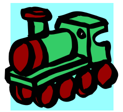
Dijkstra's Algorithm On-Line:
An Empirical Case Study from
Public Railroad Transport



Frank Schulz, Dorothea Wagner, and Karsten Weihe

University of Konstanz
Department of Computer and Information Science

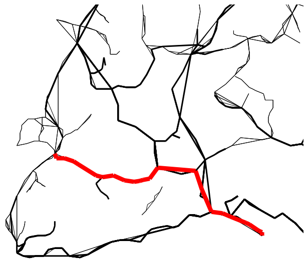
Slide 1

Introduction

Problem

- Given a set of railroad stations and a timetable.
- Find *optimal* connections between two stations.
- Example:

Konstanz	6:40	RE 18744
Donaueschingen	7:42	
Donaueschingen	7:49	RE 18705
Neustadt	8:28	
Neustadt	8:31	SE 18570
Freiburg Hbf	9:18	



Slide 2

Scenario

- Core component of an information system for public railroad transport
  - Wide-area network
  - Central information server
    ($\rightarrow$ large number of on-line queries)

$\rightarrow$ *Fast* algorithm for a low average response time

$\rightarrow$ Space consumption is less important

Slide 3

Pilot Study

- Simple queries:

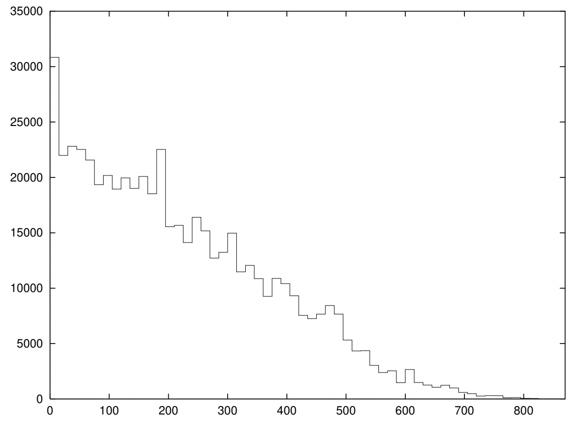
$$\textit{optimal connection} := \left\{ \begin{array}{l} \text{fastest connection subject to} \\ \text{an } \textit{earliest departure time} \end{array} \right.$$

- All trains are assumed to operate daily
- Real-world data :
  - *Train data* (winter 1996/97) of the Deutsche Bahn AG
  - Queries: a "snapshot" of the central server of the DB

Slide 4

Snapshot of Customer Queries

- Frequency distribution according to the distance



Slide 5

Model

- Definition of a *traingraph*
- Optimal connection $\leftrightarrow$ Shortest path
- Use Dijkstra's Algorithm as basis

Research Goal

- Speed-up techniques for Dijkstra's algorithm
  - *Optimality-preserving*
  - Real-world input data

Slide 6

Related Work

- Application-oriented work
  - Wide-area railroad networks: Heuristics
  - Local public transport
  - Car navigation
  - Hierarchical graph models
- Theoretical work: Dijkstra's Algorithm
  - Problem classification: single source - some targets
  - Speed-up techniques
  - Data structures

Slide 7

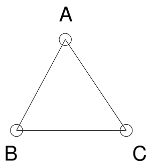
Algorithms

Definition of the Train Graph

- *Event*: *arrival* or *departure* of a train at a station
- *Nodes:* one node for every event
- *Edges:* (v, w) is a (directed) edge, if
  1. v is a *departure* of a train and w the *very next arrival*, or
  2. w is the *direct successor* of v in positive time direction at the *same station*
- Length of an edge: Time difference between endnodes

Slide 8

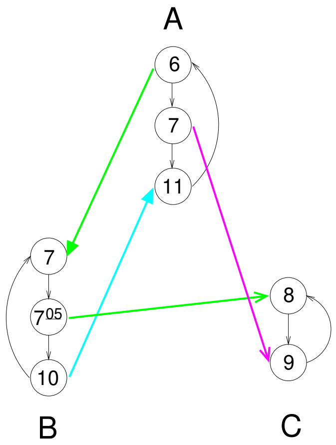
Simple Example of a Train Graph



Train I		
leaving	A	6
arriving	B	7
leaving	B	7:05
arriving	C	8

Train II		
leaving	B	10
arriving	A	11

Train III		
leaving	A	7
arriving	C	9



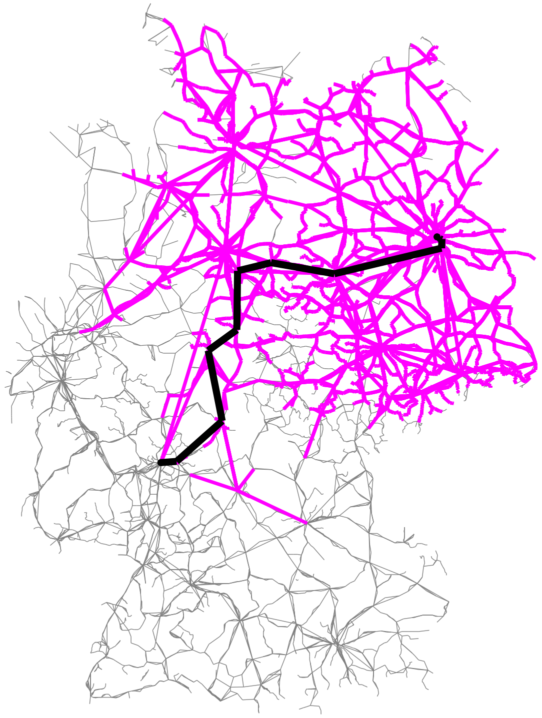
Slide 9

Dijkstra's Algorithm - Common Speed-up Techniques

- Priority queue: *Heap* and *Dial* implementation
- Restriction of the search horizon
- Avoidance of initialization for every query
- Goal-directed search
- Not directly applicable: Bidirectional search
  → lower bound on the "real" runtime using an idealized variant

Slide 10

Dijkstra's Algorithm - Sample Query



Slide 11

Speed-up Technique I: Angle Restriction

- Relies on coordinates for stations
- *Preprocessing:*
  Additional edge-information: for each edge $e = (v, w)$
  - Determine *all* shortest paths $\mathcal{P}$ from v to *all* stations
  - Store condensed information: A *circle sector* $\Lambda = \Lambda(e)$ with center at v:

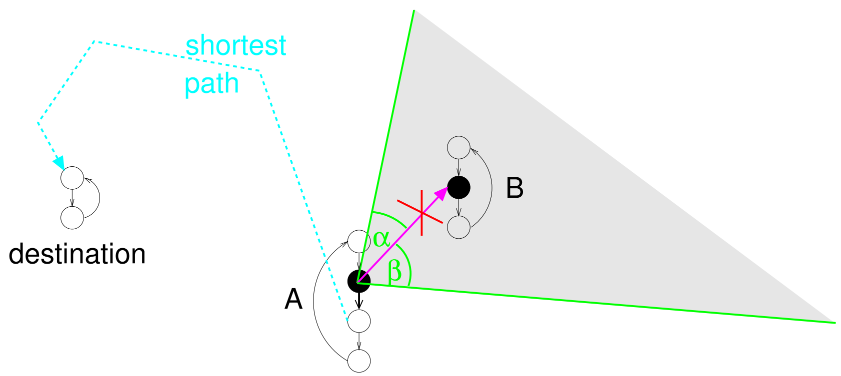
$$\forall p \in \mathcal{P}: \; p = (v, w, \ldots, d) \rightarrow d \in \Lambda$$

- Consequence for an *on-line query* from s to D:

  Ignore an edge e in Dijkstra's algorithm if $D \notin \Lambda(e)$

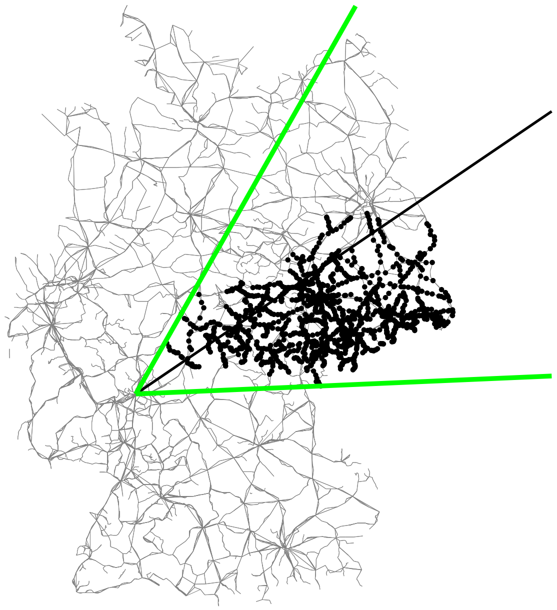
Slide 12

Angle Restriction - Idea



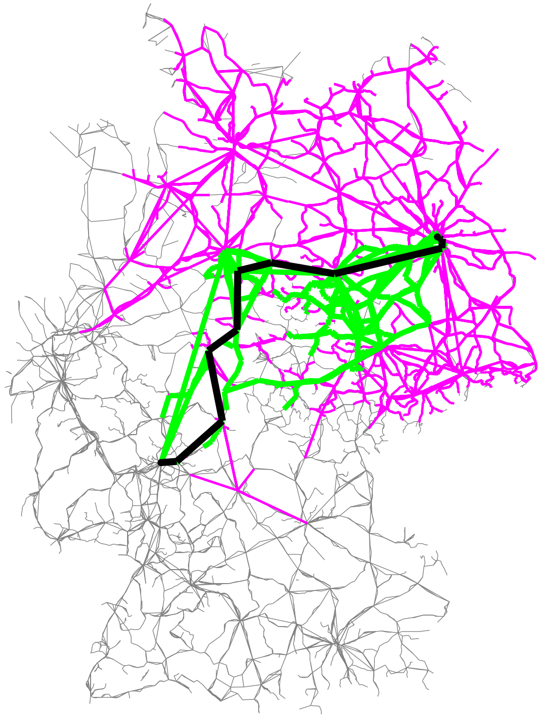
Slide 13

Angle Restriction - Circle Sector



Slide 14

Angle Restriction - Sample Query

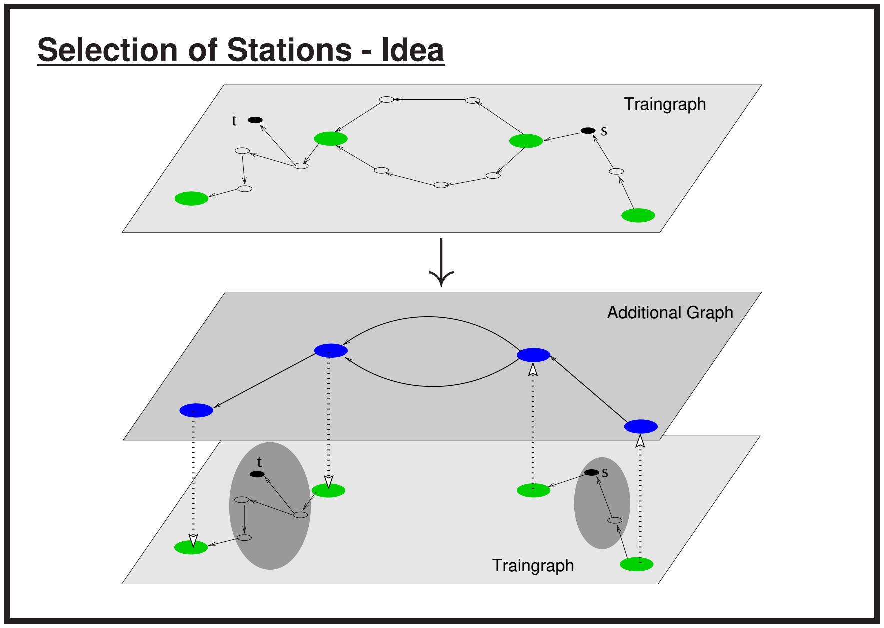


Slide 15

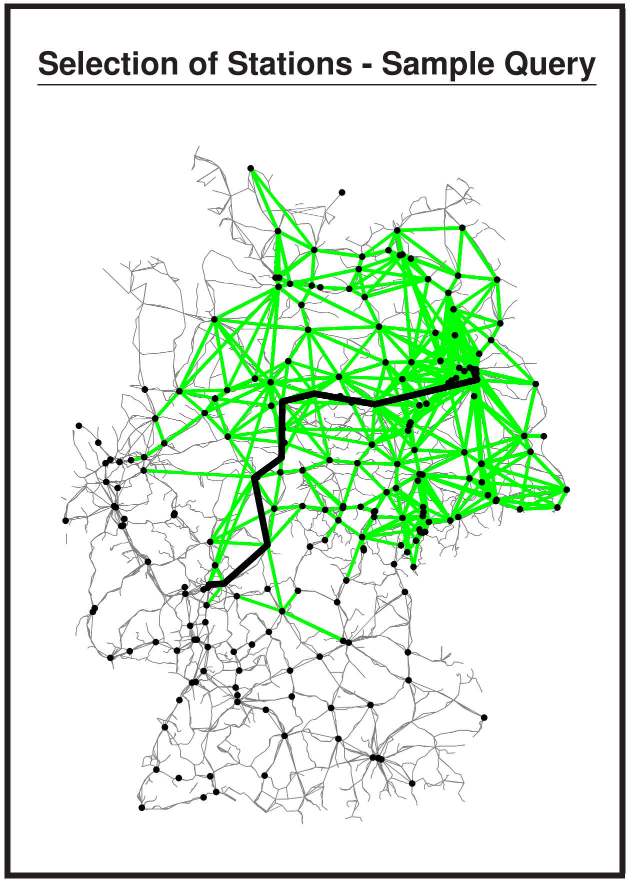
Speed-up Technique II: Selection of Stations

- Small set of selected stations
- Auxiliary graph:
  - *Nodes:* one node for every event on the selected stations
  - *Edges:* (v, w) is an edge iff there is a path from v to w such that no internal node belongs to a selected station

Slide 16

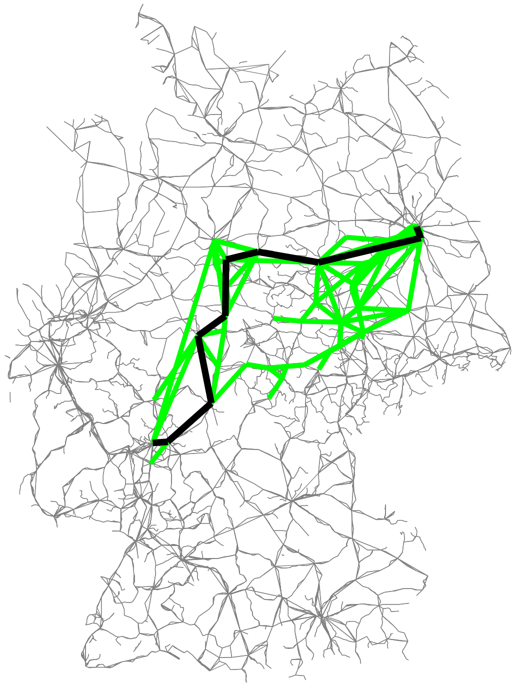


Slide 17



Slide 18

Combination of Techniques I & II



Slide 19

Analysis of Performance

Average CPU time

Selection of stations	Angle restriction	CPU heap	CPU dial
no	no	0.310	0.103
no	yes	0.036	0.018
yes	no	0.027	0.012
yes	yes	0.005	0.003

Slide 20

Analysis of Performance

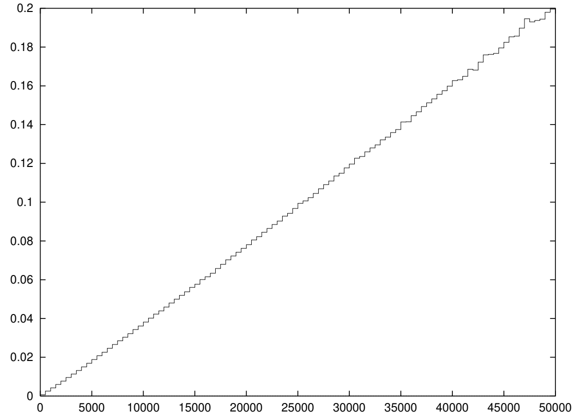
Relative Performance

Speed-up technique	Nodes	CPU-time (Dial)
-	1.00	1.00
Goal-directed search	0.63	0.70
Bidirected search	0.32	-
Angle Restriction	0.27	0.17
Selection of Stations	0.12	0.12
Combination	0.04	0.03

Slide 21

Linearity of CPU Time in the Number of Nodes

- Touched nodes $\leftrightarrow$ CPU-time



Slide 22

Final Remarks

Conclusion

- Success for geometric speed-up techniques
- Linear behaviour of runtime
- Compromise: performance $\leftrightarrow$ waste of other resources

Outlook

- Trains don't operate daily
- Extended queries: further criteria and restrictions

$\rightarrow$ Preprocessing has to be modified

Slide 23