Karlsruher Institut für Technologie

Algorithmen für Routenplanung

13. Sitzung, Sommersemester 2014

Prof. Christos Zaroliagis | 28. Mai 2014

Institut für Theoretische Informatik · Algorithmik · Prof. Dr. Dorothea Wagner

Computing Alternative Routes

Motivation

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Computing Alternative Routes

Motivation

- Humans ...

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Computing Alternative Routes

Karlsruher Institut für Technologie

Motivation

- Humans ...
  - strange creatures

Computing Alternative Routes

Motivation

- Humans ...
  - strange creatures
    have different preferences and criteria, not easily quantifiable
    e.g., like/dislike certain route parts

Computing Alternative Routes

Motivation

- Humans ...
  - strange creatures
    have different preferences and criteria, not easily quantifiable
    e.g., like/dislike certain route parts
  - typically thankless

Computing Alternative Routes

Motivation

- Humans ...
  - strange creatures
    have different preferences and criteria, not easily quantifiable
    e.g., like/dislike certain route parts
  - typically thankless
    love **choices** (even if they are provided with an optimal route)

Computing Alternative Routes

Motivation

- Humans ...
  - strange creatures
    have different preferences and criteria, not easily quantifiable
    e.g., like/dislike certain route parts
  - typically thankless
    love **choices** (even if they are provided with an optimal route)
  - back-up choices for altering a route
    in case of emergent traffic conditions (jams, construction works, etc)

Computing Alternative Routes

Karlsruher Institut für Technologie

Motivation

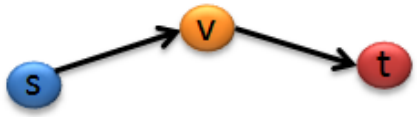
- Humans ...
  - strange creatures
    have different preferences and criteria, not easily quantifiable
    e.g., like/dislike certain route parts
  - typically thankless
    love **choices** (even if they are provided with an optimal route)
  - back-up choices for altering a route
    in case of emergent traffic conditions (jams, construction works, etc)

Goal – computing alternative routes

- Provide good/reasonable alternatives
  non-overlapping paths with optimal or near optimal cost

Existing Approaches

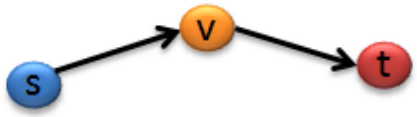
- **Via Node**
  - *s*-*v*-*t* paths : *s*-*v* and *v*-*t* shortest paths



  - restricted tracing/selection under certain local optimality criteria
  - few alternatives (not always successful)

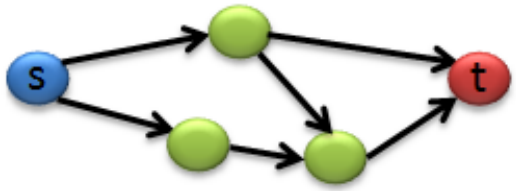
Existing Approaches

- **Via Node**
  - *s*-*v*-*t* paths : *s*-*v* and *v*-*t* shortest paths



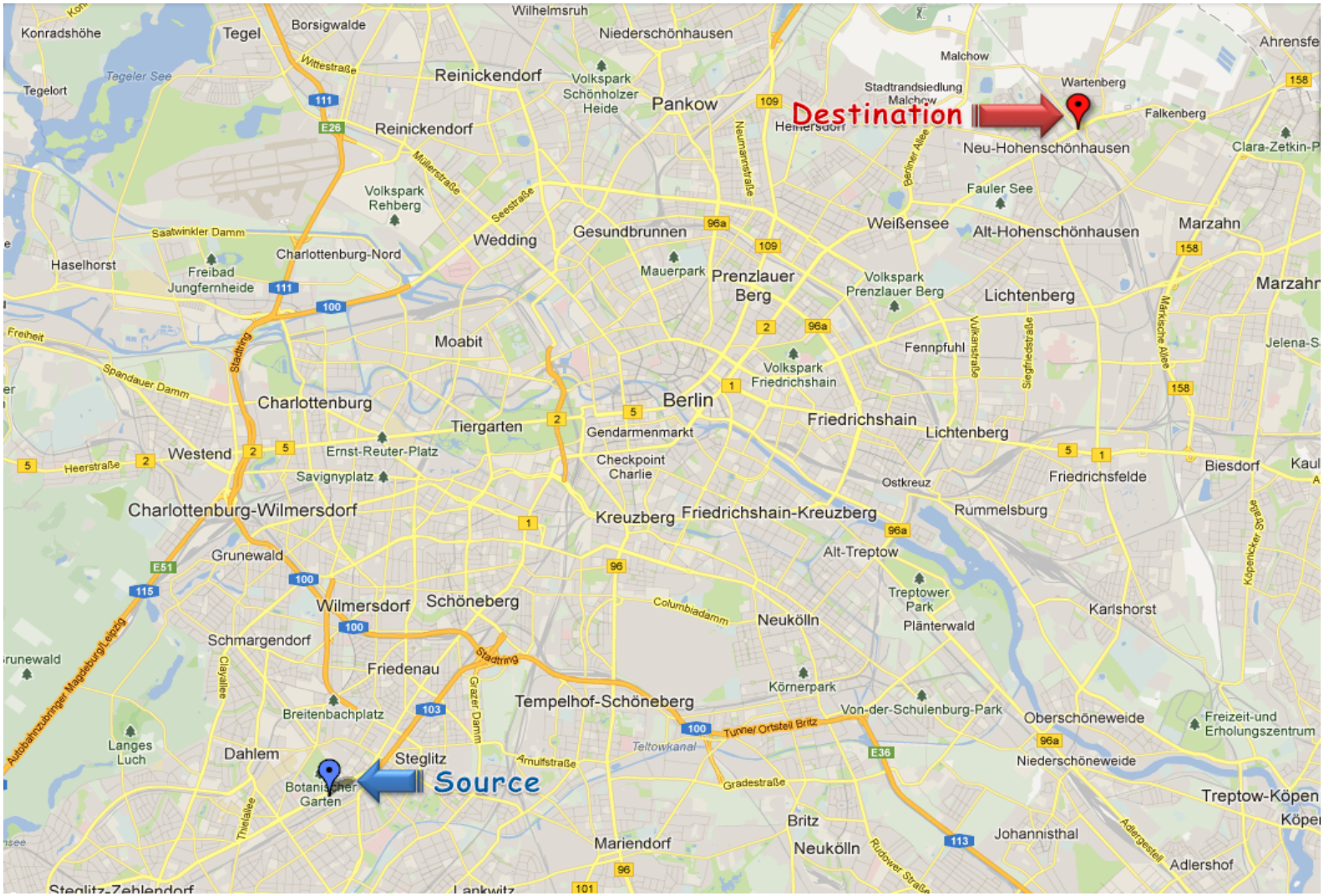
  - restricted tracing/selection under certain local optimality criteria
  - few alternatives (not always successful)

- **Alternative Graph**



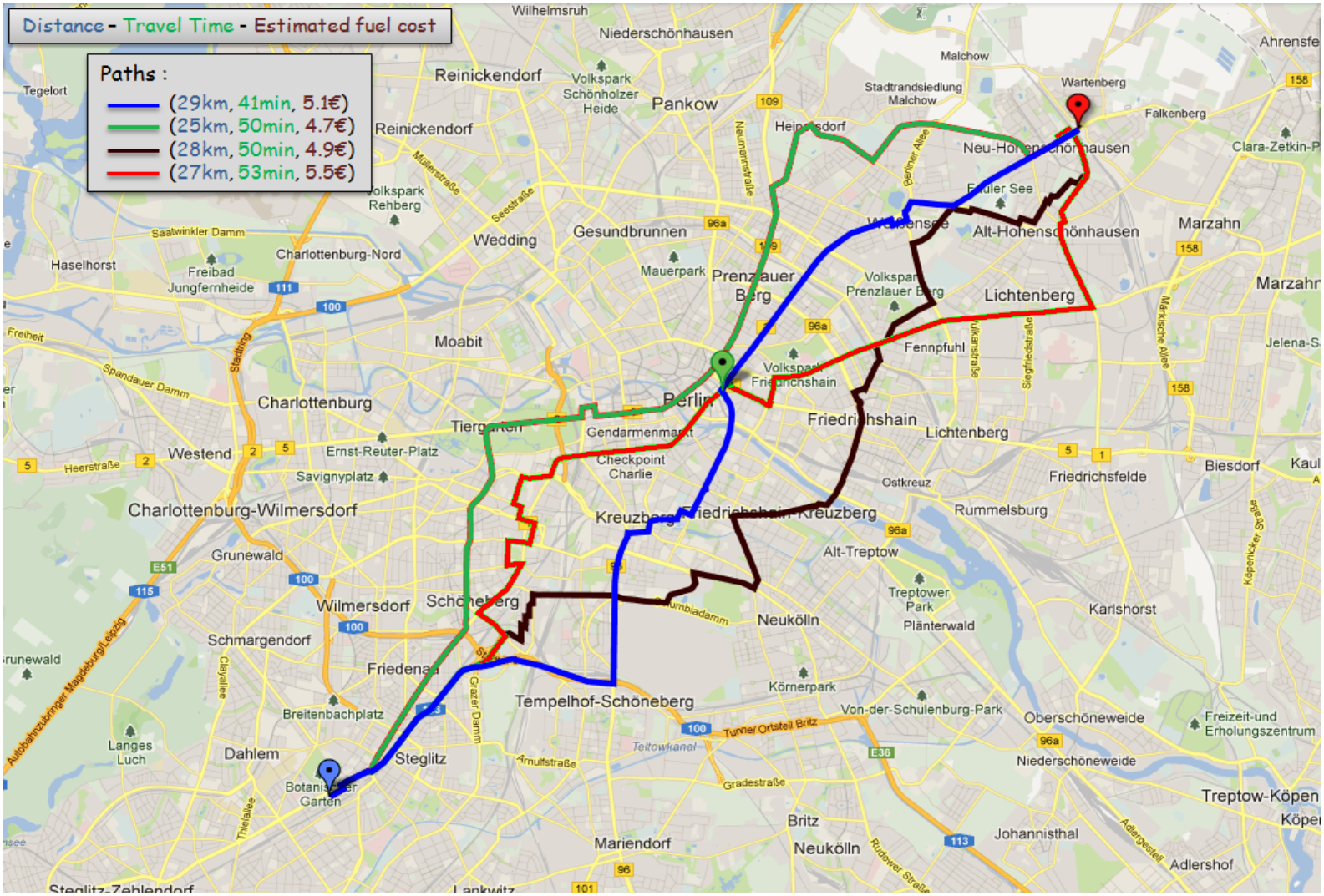
  - AG : ∪ *s*-*t* paths
  - unrestricted tracing/selection
  - many qualitative alternatives
  - more suitable for practical navigation systems

Alternative Graph - Motivation



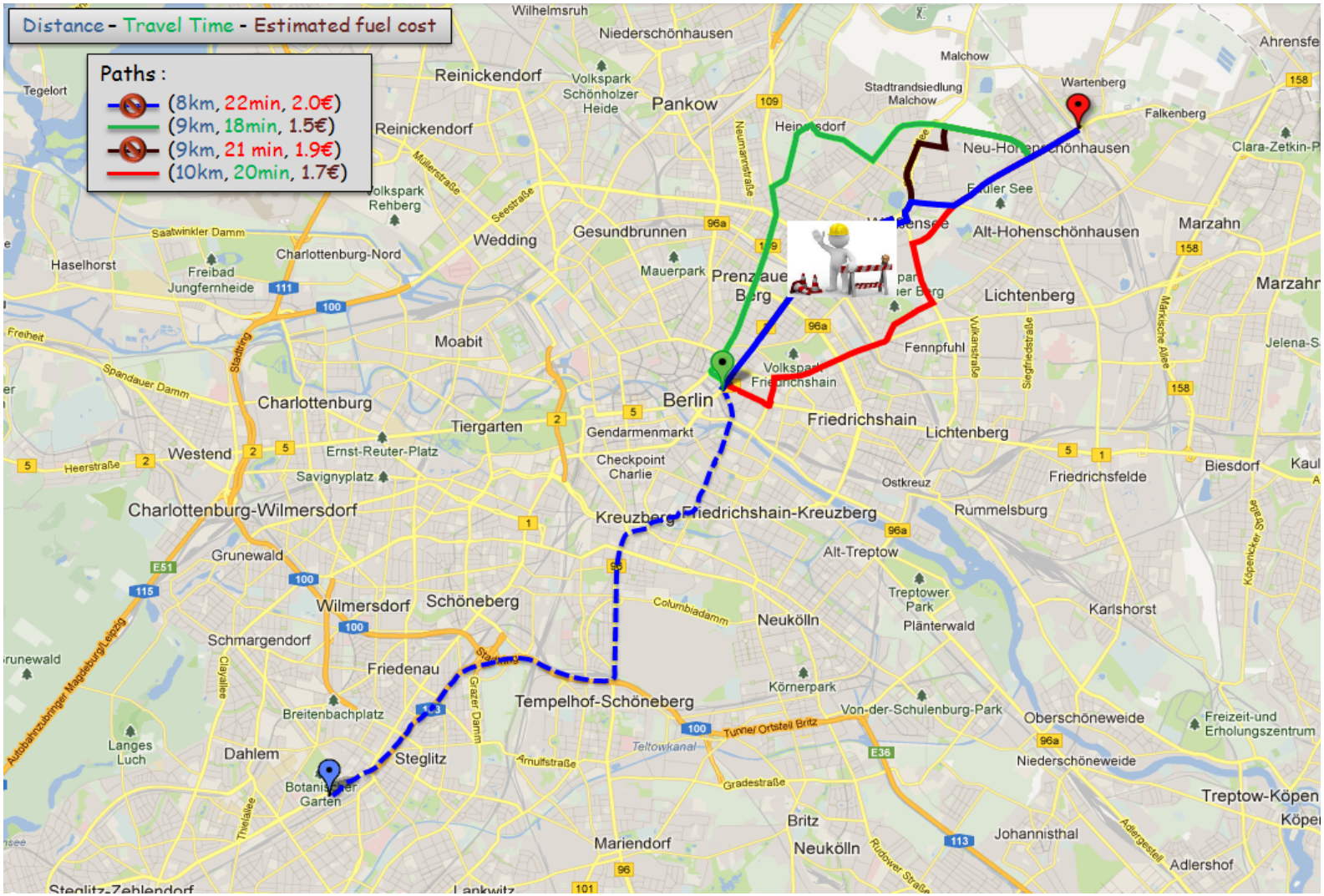
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Alternative Graph - Motivation



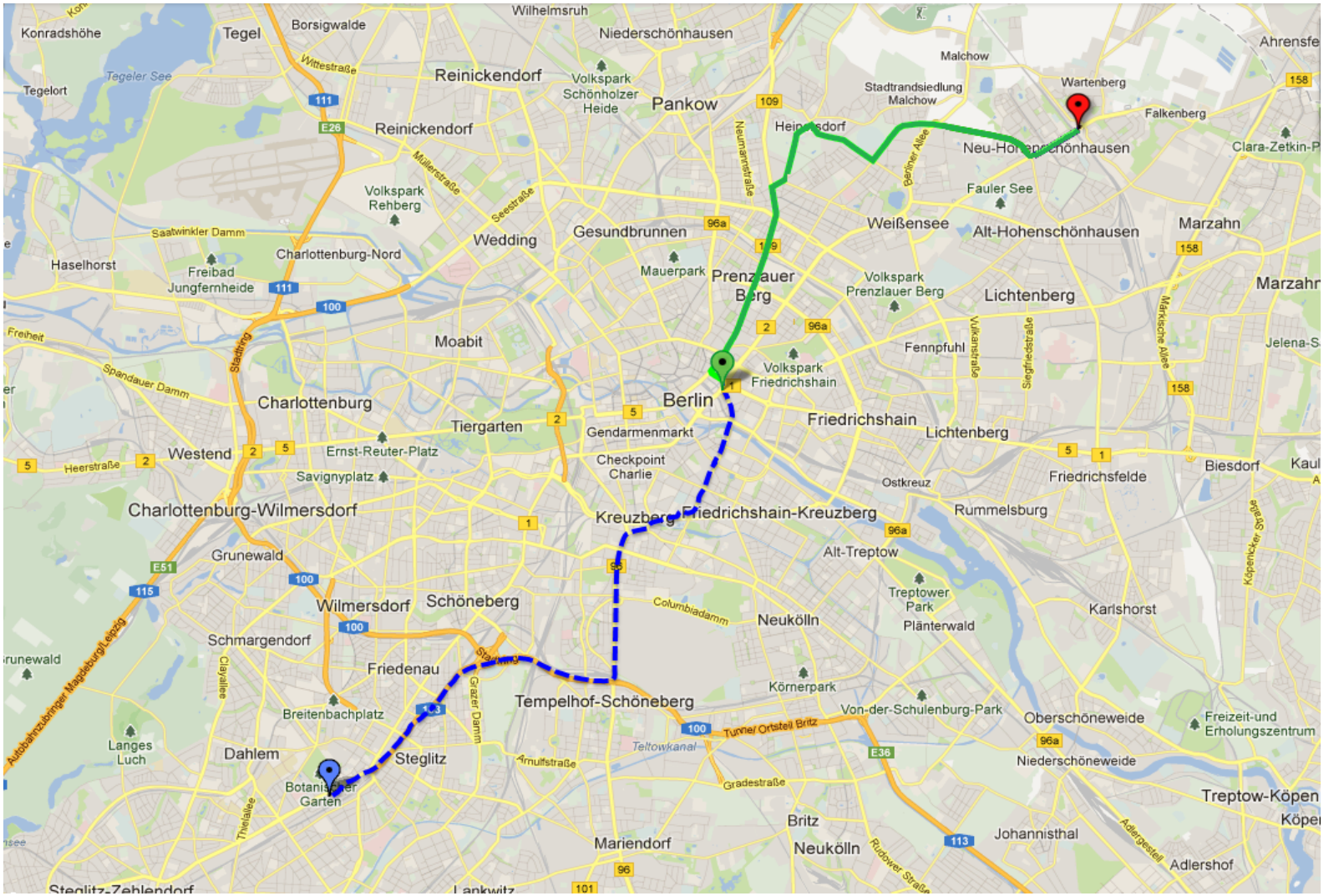
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Alternative Graph - Motivation

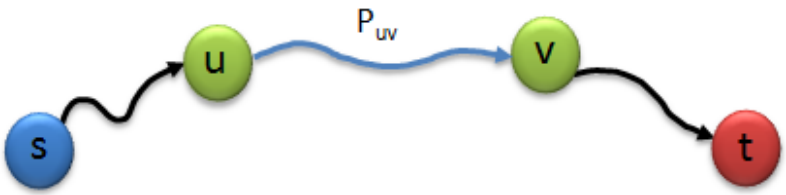


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

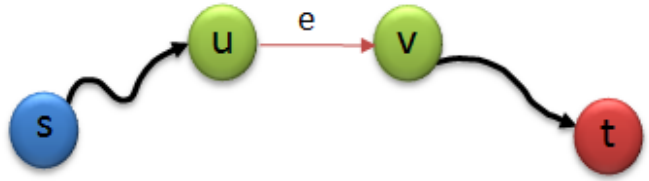
Alternative Graph - Motivation



Alternative Graph



$G(V, E)$: Original graph



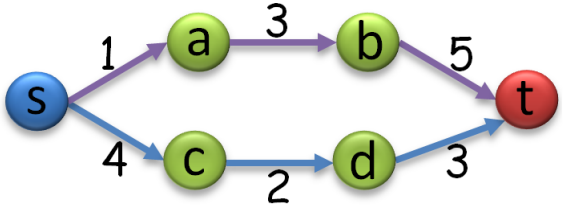
$H(V', E')$: Alternative Graph

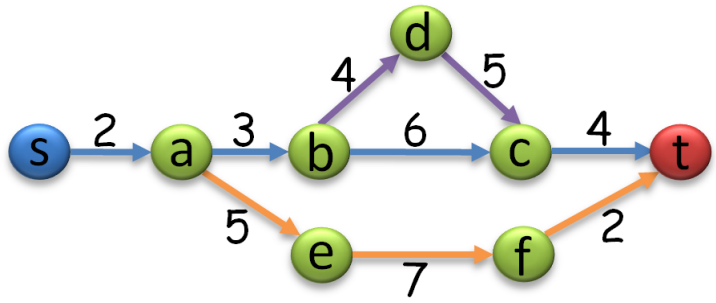
- AG $H(V', E')$: $\cup$ s-t paths
- $V' \subseteq V$
- $\forall e = (u, v) \in E' \Rightarrow \exists$ path P_{st} in H and $e \in P_{st}$
- $\forall e \in P_{st} \Rightarrow \exists P_{uv}$ in G : $w_H(e) = w_G(P_{uv})$

Quality indicators

d_H : shortest distance in graph H

$$totalDistance = \sum_{e=(u,v)\in E'} \frac{w(e)}{d_H(s,u) + w(e) + d_H(v,t)} \qquad (overlapping)$$





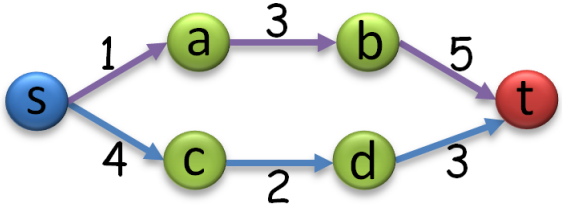
$$totalDistance = \frac{1+3+5}{9} + \frac{4+2+3}{9} = 2$$

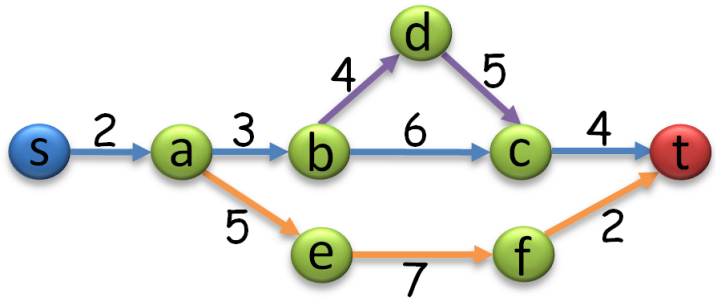
$$totalDistance = \frac{2+3+6+4}{15} + \frac{4+5}{18} + \frac{5+7+2}{16}$$
$$= 1 + 0.5 + 0.875 = 2.375$$

Quality indicators

d_H : shortest distance in graph H

$$averageDistance = \frac{\sum_{e \in E'} w(e)}{d_G(s,t) \cdot totalDistance} \qquad (stretch)$$





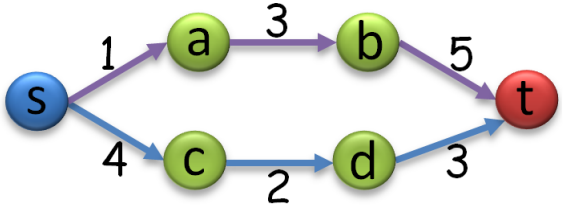
$$averageDistance = \frac{18}{9 \times 2} = 1$$

$$averageDistance = \frac{1 \times 15 + 0.5 \times 18 + 0.875 \times 16}{(1 + 0.5 + 0.875) \times 15} = 1.067$$

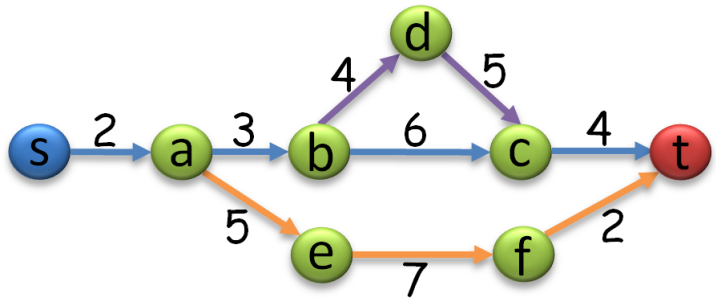
Quality indicators

d_H : shortest distance in graph H

$$decisionEdges = \sum_{v \in V' \setminus \{t\}} (outdegree(v) - 1) \qquad (size\ of\ AG)$$



decisionEdges = 1



decisionEdges = 2

Target Function

$$targetFunction = \max\{totalDistance - averageDistance\} + 1$$

bounds

- $averageDistance \leq 1.1$
  alternatives at most 10% larger than the shortest route

- $decisionEdges \leq 11$
  small in order to acquire a clearly readable AG

Target Function

$$targetFunction = \max\{totalDistance - averageDistance\} + 1$$

bounds

- $averageDistance \leq 1.1$
  alternatives at most 10% larger than the shortest route

- $decisionEdges \leq 11$
  small in order to acquire a clearly readable AG

Complexity:

Target Function

$$targetFunction = \max\{totalDistance - averageDistance\} + 1$$

bounds

- $averageDistance \leq 1.1$
  alternatives at most 10% larger than the shortest route

- $decisionEdges \leq 11$
  small in order to acquire a clearly readable AG

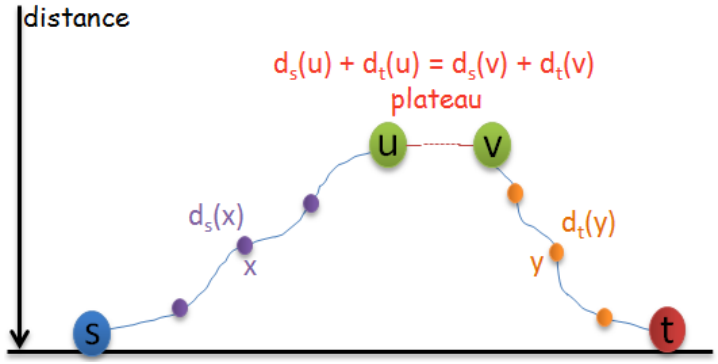
Complexity: NP-hard

Approaches for Computing AGs

- Plateau
- Penalty

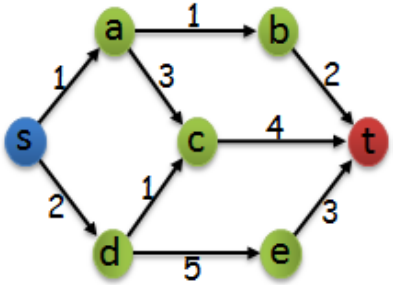
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Approaches for Computing AGs – Plateau

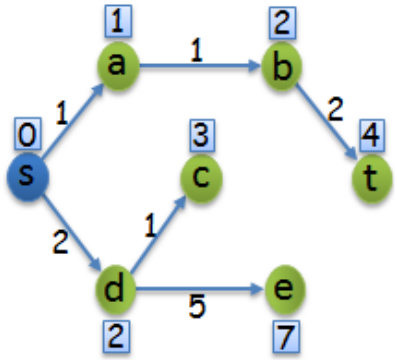


$d_s(v) \equiv d(s, v)$
$d_t(v) \equiv d(v, t)$

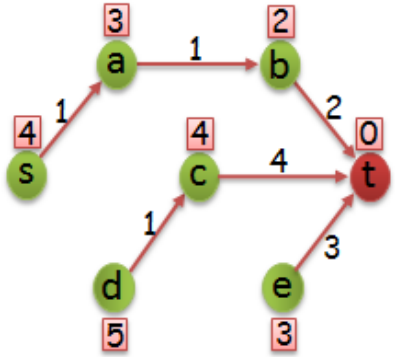
- Plateau $\overline{P} \subseteq P_{st} : \forall u, v \in \overline{P} \;\; d_s(u) + d_t(u) = d_s(v) + d_t(v)$
- Alternative paths $\Leftarrow$ combined s-v and v-t shortest paths : $v \in \overline{P}$



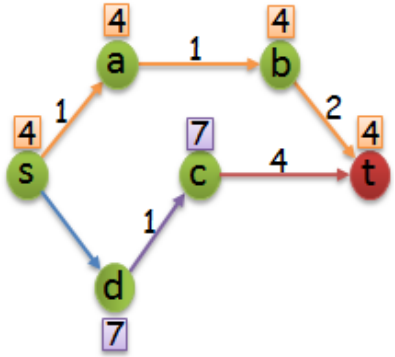
graph G



forward shortest path tree



backward shortest path tree



plateaus: s-a-b-t and d-c

Approaches for Computing AGs – Penalty

- Compute a shortest path P_{st}

Approaches for Computing AGs – Penalty

- Compute a shortest path P_{st}
- Penalize P_{st}: increase the weight of its edges

Approaches for Computing AGs – Penalty

- Compute a shortest path P_{st}
- Penalize P_{st}: increase the weight of its edges
- Prevent small detours: penalize edges leaving and re-joining AG

Approaches for Computing AGs – Penalty

- Compute a shortest path P_{st}
- Penalize P_{st}: increase the weight of its edges
- Prevent small detours: penalize edges leaving and re-joining AG
- Compute a new P'_{st}; if P'_{st} is short and different enough from previous P_{st}, then add it to AG

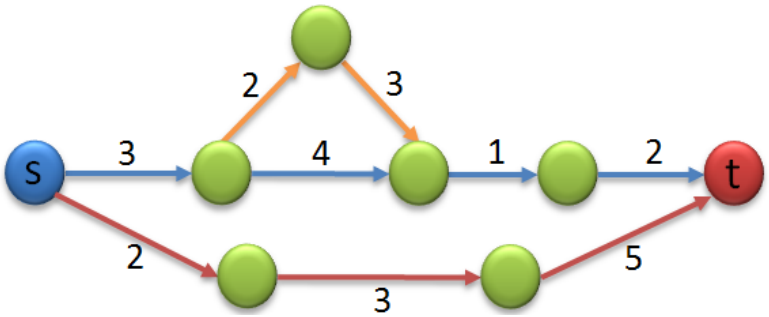
Approaches for Computing AGs – Penalty

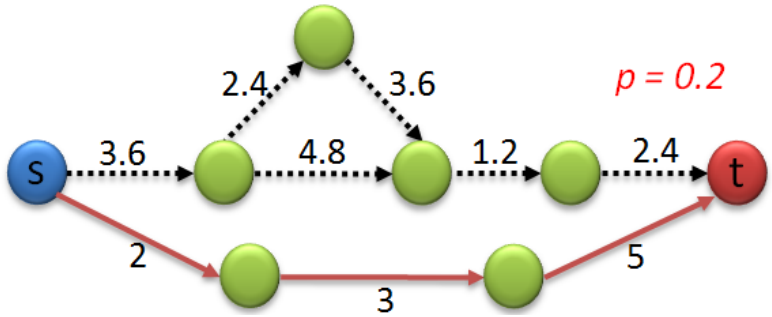
- Compute a shortest path P_{st}
- Penalize P_{st}: increase the weight of its edges
- Prevent small detours: penalize edges leaving and re-joining AG
- Compute a new P'_{st}; if P'_{st} is short and different enough from previous P_{st}, then add it to AG
- Repeat until a sufficient number of alternatives is found, or weight adjustments of s-t paths bring no better results

Approaches for Computing AGs – Penalty

- $w(e) = w(e) + p \cdot w_{orig}(e)$
- penalty factor $p \in (0, \infty)$ over P_{st} edges e
  typically $0.1 \leq p \leq 1$

- $w(e) = w(e) + r \cdot w_{orig}(e)$
- rejoin-penalty factor $r \in (0, \infty)$ over adjacent to P_{st} edges e
  typically $0.1 \leq r \leq 1$





An Improved Method

- Search pruning
  - locate shortest *s*-*t* paths faster
- Filtering and fine-tuning
  - improve Penalty and Plateau methods
  - obtain best alternative *s*-*t* paths
    (minimum stretch and overlapping)

Search Pruner

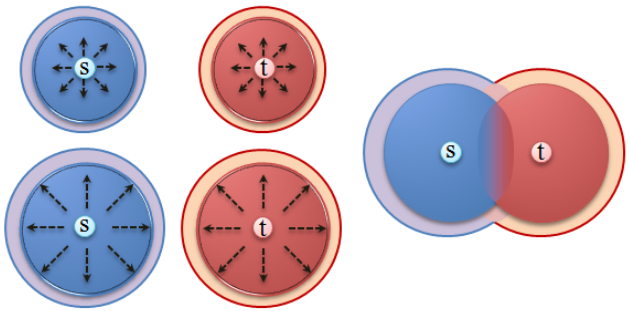
Algorithms

- Uninformed Bidirectional Pruner
  without preprocessing
- Informed ALT Bidirectional Pruner
  (A*+landmarks+triangle inequality)
  preprocessing (shortest distances from/to landmarks)

Results

- detect faster the nodes on s-t shortest paths
- reduce search space
  discard nodes $v : d_s(v) + d_t(v) > \tau \cdot d_s(t), \tau \in [1, 2]$ (max stretch)

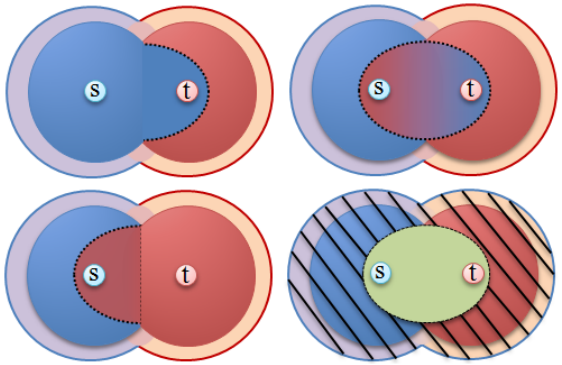
Search Pruner



searches meet each other, $d_s(t)$ is traced

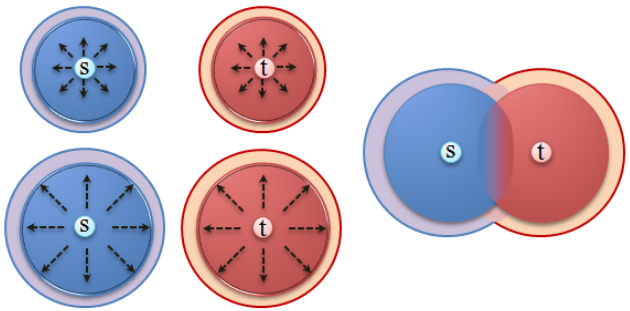
- $h_s(v)$: lower bound on $d_s(v) \equiv d(s, v)$
- $h_t(v)$: lower bound on $d_t(v) \equiv d(v, t)$

- Forward and backward search discard node v:
  $d_s(v) + h_t(v) > \tau \cdot d_s(t)$
  $h_s(v) + d_t(v) > \tau \cdot d_s(t)$

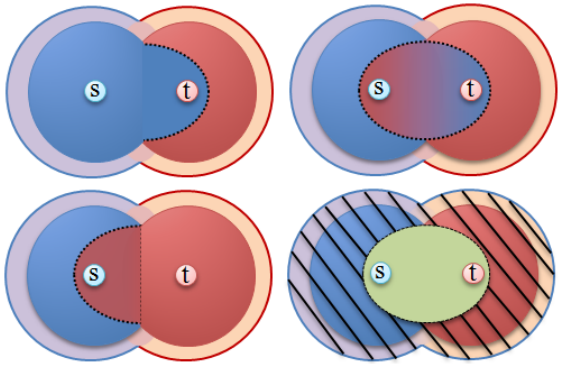


locate nodes in shortest s-t paths

Search Pruner



searches meet each other, $d_s(t)$ is traced



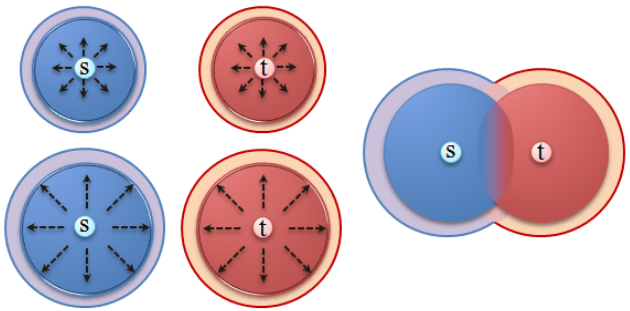
locate nodes in shortest s-t paths

- $h_s(v)$: lower bound on $d_s(v) \equiv d(s, v)$
- $h_t(v)$: lower bound on $d_t(v) \equiv d(v, t)$

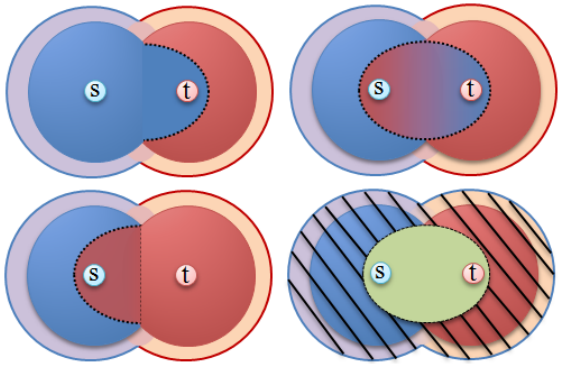
- Forward and backward search discard node v:
  $d_s(v) + h_t(v) > \tau \cdot d_s(t)$
  $h_s(v) + d_t(v) > \tau \cdot d_s(t)$

- Uninformed Bidirectional Pruner
  based on forward and backward search queues
  $d_s(v) \geq h_s(v) = Q_f.minKey()$
  $d_t(v) \geq h_t(v) = Q_b.minKey()$
  slow bounding

Search Pruner



searches meet each other, $d_s(t)$ is traced



locate nodes in shortest s-t paths

- $h_s(v)$: lower bound on $d_s(v) \equiv d(s, v)$
- $h_t(v)$: lower bound on $d_t(v) \equiv d(v, t)$

- Forward and backward search discard node v:
  $d_s(v) + h_t(v) > \tau \cdot d_s(t)$
  $h_s(v) + d_t(v) > \tau \cdot d_s(t)$

- **Uninformed Bidirectional Pruner**
  based on forward and backward search queues
  $d_s(v) \geq h_s(v) = Q_f.minKey()$
  $d_t(v) \geq h_t(v) = Q_b.minKey()$
  slow bounding

- **Informed ALT Bidirectional Pruner**
  based on ALT heuristics
  $h_s(v) =$
  $\max_L\{d(L, v) - d(L, s), d(s, L) - d(v, L)\}$
  $h_t(v) = \max_L\{d(L, t) - d(L, v), d(v, L) - d(t, L)\}$
  fast bounding

Filtering

New ranking of plateaus $\Rightarrow$ obtain better alternative paths

x-y plateau $\overline{P}$:

$rank = totalDistance - averageDistance$, (vs $w(P_{st}) - w(\overline{P})$)

$totalDistance = \frac{w(\overline{P})}{d_s(x)+w(\overline{P})+d_t(y)}$ (overlapping)

$averageDistance = \frac{w(\overline{P})+d_s(t)}{(1+totalDistance)\cdot d_s(t)}$ (stretch)

Fine-tuning

New adjustment of weight increases in adjacent edges of P_{st}

- $w(e) = w(e) + (0.1 + r \cdot d_s(u)/d_s(t)) \cdot w_{orig}(e), \forall e = (u,v) \in E : \ u \in P_{st}, \ v \notin P_{st}$
  heavier weights on outgoing edges closer to t
- $w(e) = w(e) + (0.1 + r \cdot d_t(v)/d_t(s)) \cdot w_{orig}(e), \forall e = (u,v) \in E : \ u \notin P_{st}, \ v \in P_{st}$
  heavier weights on incoming edges closer to s
- slightly heavier penalty on edges $e = (u,v) \in P_{st}$, $outdegree(u) > 1$

▶ prevents recomputing already traced alternative paths

+ online checking of quality indicators, discarding paths that do not improve them

Combination

Combined Plateau and Penalty

- same pruning stage
- extend the number of decision edges (thinout *AG* at the end)
- set penalty on the paths stored by Plateau in *AG*
- collect the best alternatives from Penalty and Plateau
  $\Rightarrow$ (maximize targetFunction)

Experimental Evaluation

- Intel(R) Xeon(R) X3430 @ 2.40GHz, 32Gb RAM
- C++, GCC-4.6.3
- European road networks (OSM)
- Berlin metropolitan area (TomTom)

map		*n*	*m*
B	Berlin	117,839	310,152
LU	Luxembourg	51,576	119,711
BE	Belgium	576,465	1,376,142
IT	Italy	2,425,667	5,551,700
GB	GreatBritain	3,233,096	7,151,300
FR	France	4,773,488	11,269,569
GE	Germany	7,782,773	18,983,043
WE	WesternEurope	26,498,732	62,348,328

- Quality indicators:
  targetFunction(*TargFun*), totalDistance(*TotDist*),
  averageDistance(*AvgDist*) and decisionEdges(*DecEdges*)

Results

- $\tau \in [1, 1.1] \wedge dist < 300km$ on large networks (GE, WE)
- $\tau = 1.2$ on small networks

map	TargFun		TotDist	AvgDist	DecEdges	Time
	(max:11)	best previous	(max:11)	(min:1)	(max:10)	(ms)
B	3.82	-	3.91	1.09	9.95	45.61
LU	4.44	3.05	4.49	1.05	9.73	37.05
BE	4.83	-	4.87	1.04	10.00	85.08
IT	4.10	-	4.14	1.04	9.92	114.29
GB	4.36	-	4.40	1.04	9.93	180.12
FR	4.22	-	4.26	1.04	9.97	159.93
GE	4.88	-	4.92	1.04	10.00	286.40
WE	4.35	3.08	4.37	1.02	9.88	717.57

- $|AG| \leq c \cdot |SP_{st}|$, typically $c \in [2, 3]$

Results

- $\tau \in [1, 1.1] \wedge dist < 300km$ on large networks (GE, WE)
- $\tau = 1.2$ on small networks

map	TargFun		TotDist	AvgDist	DecEdges	Time
	(max:11)	best previous	(max:11)	(min:1)	(max:10)	(ms)
B	4.16	-	4.23	1.07	9.92	49.34
LU	5.14	2.91	5.19	1.05	9.23	41.56
BE	5.29	-	5.33	1.04	9.54	159.71
IT	4.11	-	4.14	1.03	9.47	105.84
GB	4.38	-	4.41	1.03	9.87	210.94
FR	4.11	-	4.16	1.05	9.32	192.44
GE	5.42	-	5.46	1.04	9.91	388.97
WE	5.21	3.34	5.24	1.03	9.67	776.97

- $|AG| \leq c \cdot |SP_{st}|$, typically $c \in [2, 3]$

Results

- $\tau \in [1, 1.1] \wedge dist < 300km$ on large networks (GE, WE)
- $\tau = 1.2$ on small networks

map	TargFun		TotDist	AvgDist	DecEdges	Time
	(max:11)	best previous	(max:11)	(min:1)	(max:10)	(ms)
B	4.55	-	4.61	1.06	9.97	54.12
LU	5.25	3.29	5.30	1.05	9.81	43.69
BE	5.36	-	5.41	1.05	9.89	163.75
IT	4.37	-	4.41	1.04	9.79	178.08
GB	4.67	-	4.71	1.04	9.86	284.38
FR	4.56	-	4.60	1.04	9.86	217.30
GE	5.50	-	5.54	1.04	9.89	446.38
WE	5.49	3.70	5.52	1.03	9.94	987.42

- $|AG| \leq c \cdot |SP_{st}|$, typically $c \in [2, 3]$

Results

- $\tau = 1.2 \wedge dist < 500km$

map WE	TargFun	TotDist	AvgDist	DecEdges
Plateau	4.71	4.73	1.02	10.00
Penalty	6.46	6.48	1.02	9.97
Plateau & Penalty	6.82	6.84	1.02	9.98

- $|AG| \leq c \cdot |SP_{st}|$, typically $c \in [2, 3]$

Conclusion

- Extended and improved versions of Penalty and Plateau methods
- Large number of qualitative alternatives with
  a) high non-overlappingness
  b) low stretch
- Time execution: less than 1 second on continental size networks
- ALT-based heuristics beneficial: resistant to edge cost increases
  ($\nexists$ need for new preprocessing)

Literatur

Alternativgraphen:

- Roland Bader, Jonathan Dees, Robert Geisberger, Peter Sanders
  Alternative Route Graphs in Road Networks
  In: *Proceedings of the 1st International ICST Conference on Theory and Practice of Algorithms in (Computer) Systems (TAPAS'11)*, 2011
- Andreas Paraskevopoulos, Christos Zaroliagis
  Improved Alternative Route Planning
  In: *Proceedings of the 13th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'13)*, 2013

Nächste Termine

Karlsruher Institut für Technologie

Montag, 2.6.2014

Montag, 11.6.2014
(Auf englisch)
Mittwoch, 18.6.2014
(Auf englisch)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik