Karlsruhe Institute of Technology

Evolution and Evaluation of the Penalty Method for Alternative Graphs

Moritz Kobitzsch, Marcel Radermacher, and Dennis Schieferdecker
{kobitzsch,schieferdecker}@kit.edu, marcel.radermacher@student.kit.edu

Institute of Theoretical Informatics - Algorithmics

Motivation

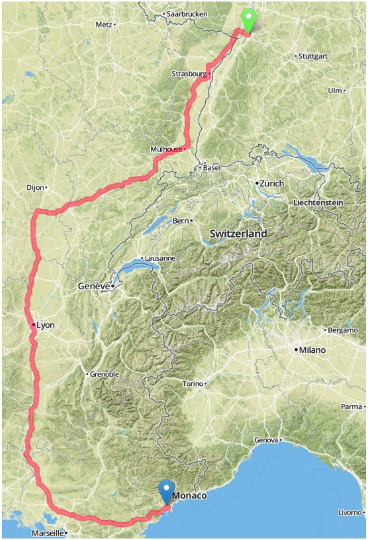
advanced route planning

shortest paths

- multitude of speed-up techniques
  (AF, CH, HL, . . . ; sub-microsecond queries)

alternative routes

- non-optimal routes
  (heuristiccs; quality $\Longleftrightarrow$ speed)
- speed-up techniques do not work well
  (algorithms do not use or relax them)



Motivation

advanced route planning

shortest paths

- multitude of speed-up techniques
  (AF, CH, HL, . . . ; sub-microsecond queries)

alternative routes

- non-optimal routes
  (heuristiccs; quality $\Longleftrightarrow$ speed)
- speed-up techniques do not work well
  (algorithms do not use or relax them)

Kobitzsch, Radermacher, Schieferdecker:
Evolution and Evaluation of the Penalty Method for Alternative Graphs

Institute of Theoretical Informatics
Algorithmics

Motivation

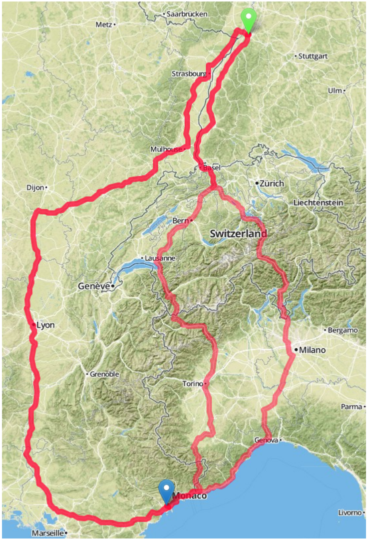
advanced route planning

shortest paths

- multitude of speed-up techniques
  (AF, CH, HL, . . . ; sub-microsecond queries)

alternative routes

- non-optimal routes
  (heuristiccs; quality $\Longleftrightarrow$ speed)
- speed-up techniques do not work well
  (algorithms do not use or relax them)



Motivation

advanced route planning

shortest paths

- multitude of speed-up techniques
  (AF, CH, HL, . . . ; sub-microsecond queries)

alternative routes

- non-optimal routes
  (heuristiccs; quality $\Longleftrightarrow$ speed)
- speed-up techniques do not work well
  (algorithms do not use or relax them)

Motivation

what are alternative graphs?

alternative graph (AG)

- extension to alternative routes
  - ⇒ encode multiple (good) alternative routes (between one source and one target)
  - ⇒ interaction between alternative routes
  - ⇒ compact representation of options

- intermediate data structure
  - ⇒ sparse, directed graph
  - ⇒ usable with expensive algorithms (stochastic routing, traffic simulation)

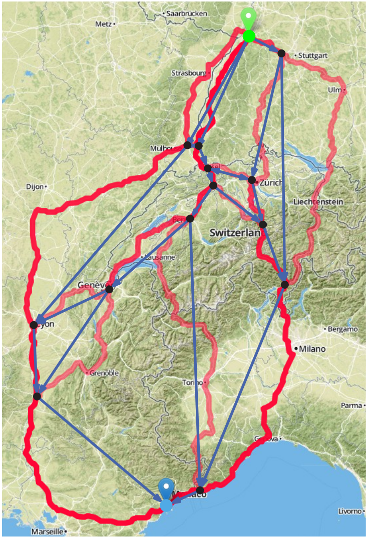
Motivation

what are alternative graphs?

alternative graph (AG)

- extension to alternative routes
  - ⇒ encode multiple (good) alternative routes (between one source and one target)
  - ⇒ interaction between alternative routes
  - ⇒ compact representation of options

- intermediate data structure
  - ⇒ sparse, directed graph
  - ⇒ usable with expensive algorithms (stochastic routing, traffic simulation)

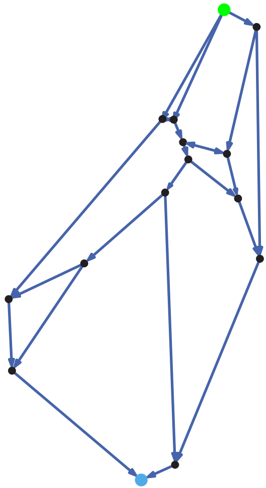


Motivation

what are alternative graphs?

alternative graph (AG)

- extension to alternative routes
  - ⇒ encode multiple (good) alternative routes
    (between one source and one target)
  - ⇒ interaction between alternative routes
  - ⇒ compact representation of options
- intermediate data structure
  - ⇒ sparse, directed graph
  - ⇒ usable with expensive algorithms
    (stochastic routing, traffic simulation)



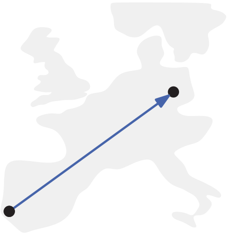
Related Work

what has been done before? (1)

"Alternative Routes in Road Networks"

[ABRAHAM ET AL. 10]

- via-node approach (plateau method)
  - → concatenation of two shortest paths
  - → variants: X-BDV (sec.), X-CHV (millisec.)



quality measures

- not too much longer (stretch)
- sufficiently different (sharing)
- reasonable (local optimality)

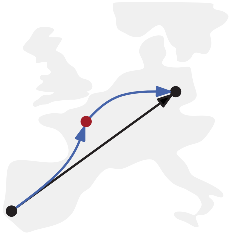
Related Work

what has been done before? (1)

"Alternative Routes in Road Networks"

[ABRAHAM ET AL. 10]

- via-node approach (plateau method)
  - → concatenation of two shortest paths
  - → variants: X-BDV (sec.), X-CHV (millisec.)



quality measures

- not too much longer (stretch)
- sufficiently different (sharing)
- reasonable (local optimality)

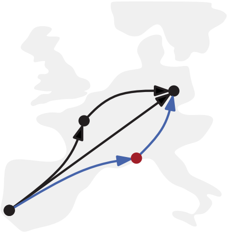
Related Work

what has been done before? (1)

"Alternative Routes in Road Networks"

[ABRAHAM ET AL. 10]

- via-node approach (plateau method)
  - → concatenation of two shortest paths
  - → variants: X-BDV (sec.), X-CHV (millisec.)



quality measures

- not too much longer (stretch)
- sufficiently different (sharing)
- reasonable (local optimality)

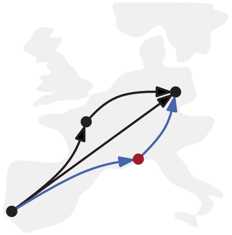
Related Work

what has been done before? (1)

"Alternative Routes in Road Networks"

[Abraham et al. 10]

- via-node approach (plateau method)
  - → concatenation of two shortest paths
  - → variants: X-BDV (sec.), X-CHV (millisec.)



quality measures

- not too much longer (stretch)
- sufficiently different (sharing)
- reasonable (local optimality)

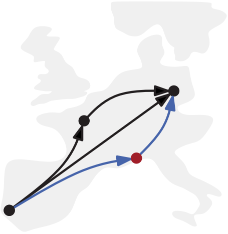
Related Work

what has been done before? (1)

"Alternative Routes in Road Networks"

[ABRAHAM ET AL. 10]

- via-node approach (plateau method)
  - → concatenation of two shortest paths
  - → variants: X-BDV (sec.), X-CHV (millisec.)



quality measures

- not too much longer (stretch)
- sufficiently different (sharing)
- reasonable (local optimality)

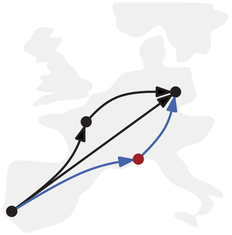
Related Work

what has been done before? (1)

"Alternative Routes in Road Networks"

[ABRAHAM ET AL. 10]

- via-node approach (plateau method)
  - → concatenation of two shortest paths
  - → variants: X-BDV (sec.), X-CHV (millisec.)



quality measures

- not too much longer (stretch)
- sufficiently different (sharing)
- reasonable (local optimality)

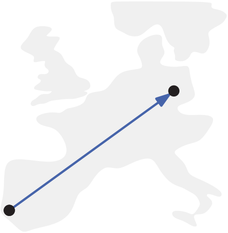
Related Work

what has been done before? (2)

"Alternative Route Graphs in Road Networks"

[BADER ET AL. 11]

- penalty method (classical method)
  - → putting penalties on arc costs
  - → based on Dijkstra (sec.)
- no alternative route extraction



quality measures

- numeric values hard to gauge
- do not directly translate to quality of alternative routes

Related Work

what has been done before? (2)

"Alternative Route Graphs in Road Networks"

[BADER ET AL. 11]

- penalty method (classical method)
  - → putting penalties on arc costs
  - → based on Dijkstra (sec.)
- no alternative route extraction

quality measures

- numeric values hard to gauge
- do not directly translate to quality of alternative routes

Related Work

what has been done before? (2)

"Alternative Route Graphs in Road Networks"

[BADER ET AL. 11]

- penalty method (classical method)
  - → putting penalties on arc costs
  - → based on Dijkstra (sec.)
- no alternative route extraction

quality measures

- numeric values hard to gauge
- do not directly translate to quality of alternative routes

Related Work

what has been done before? (2)

"Alternative Route Graphs in Road Networks"

[BADER ET AL. 11]

- penalty method (classical method)
  - → putting penalties on arc costs
  - → based on Dijkstra (sec.)
- no alternative route extraction

quality measures

- numeric values hard to gauge
- do not directly translate to quality of alternative routes

Related Work

what has been done before? (2)

"Alternative Route Graphs in Road Networks"

[BADER ET AL. 11]

- penalty method (classical method)
  - → putting penalties on arc costs
  - → based on Dijkstra (sec.)
- no alternative route extraction

quality measures

- numeric values hard to gauge
- do not directly translate to quality of alternative routes

Related Work

what has been done before? (2)

"Alternative Route Graphs in Road Networks"

[BADER ET AL. 11]

- penalty method (classical method)
  - → putting penalties on arc costs
  - → based on Dijkstra (sec.)
- no alternative route extraction

quality measures

- numeric values hard to gauge
- do not directly translate to quality of alternative routes

Related Work

what has been done before? (2)

"Alternative Route Graphs in Road Networks"

[BADER ET AL. 11]

- penalty method (classical method)
  - → putting penalties on arc costs
  - → based on Dijkstra (sec.)
- no alternative route extraction

quality measures

- numeric values hard to gauge
- do not directly translate to quality of alternative routes

Related Work

what has been done before? (2)

"Alternative Route Graphs in Road Networks"

[BADER ET AL. 11]

- penalty method (classical method)
  - → putting penalties on arc costs
  - → based on Dijkstra (sec.)
- no alternative route extraction

quality measures

- numeric values hard to gauge
- do not directly translate to quality of alternative routes

$$totalDist := \sum_{a=(u,v)\in A_H} \frac{c(a)}{\mathcal{D}_H(s,u) + c(a) + \mathcal{D}_H(v,t)}$$

$$averageDist := \frac{\sum_{a\in A_H} c(a)}{\mathcal{D}(s,t)\cdot totalDist}$$

$$\begin{aligned} \text{maximize} \quad & totalDist - (averageDist - 1) \\ \text{s.t.:} \quad & averageDist \le 1.1 \\ & decisionArcs \le 10 \end{aligned}$$

Optimization Potential

what can we do better?

current state-of-the-art

- fast "one-hop" alternative routes
- slow alternative graphs

goal

- fast alternatives with diverse structure

approach

- focus on penalty method
  - ⇒ improve for interactive use (speed-up techniques)
  - ⇒ extract alternative routes (quality criteria similar to via-node approach)
  - ⇒ analyze quality & structure of results

Penalty Method

generic workflow

```
while {termination condition false} do

    {compute shortest path}

    {add penalties to graph}

end

{select shortest paths + combine to alternative graph}

{extract alternative routes}
```

- path selection
- penalization (not covered today)

(modified from previous work)

- fast computation
- alternative route extraction

(not in previous work)

Penalty Method

path selection

basic approach (classical method)

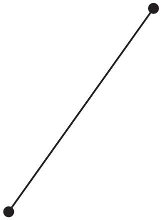
- perform "enough" iterations
  - ⇒ generate set of shortest paths
- select good subset for AG
  (implementation details left open)

when query times matter... (our approach)

- exploit quality measures
  - ⇒ terminate as paths become too long
    (allow maximum stretch w.r.t. original metric)
  - ⇒ select path for AG after each iteration
    (w.r.t. stretch, sharing measures)

Penalty Method

path selection



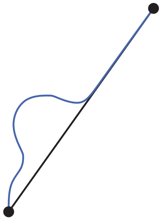
basic approach (classical method)

- perform "enough" iterations
  - ⇒ generate set of shortest paths
- select good subset for AG
  (implementation details left open)

when query times matter... (our approach)

- exploit quality measures
  - ⇒ terminate as paths become too long
    (allow maximum stretch w.r.t. original metric)
  - ⇒ select path for AG after each iteration
    (w.r.t. stretch, sharing measures)

Penalty Method
path selection



basic approach (classical method)

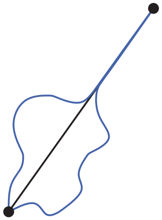
- perform "enough" iterations
  - ⇒ generate set of shortest paths
- select good subset for AG
  (implementation details left open)

when query times matter... (our approach)

- exploit quality measures
  - ⇒ terminate as paths become too long
    (allow maximum stretch w.r.t. original metric)
  - ⇒ select path for AG after each iteration
    (w.r.t. stretch, sharing measures)

Penalty Method

path selection



basic approach (classical method)

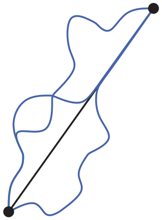
- perform "enough" iterations
  - ⇒ generate set of shortest paths
- select good subset for AG
  (implementation details left open)

when query times matter... (our approach)

- exploit quality measures
  - ⇒ terminate as paths become too long
    (allow maximum stretch w.r.t. original metric)
  - ⇒ select path for AG after each iteration
    (w.r.t. stretch, sharing measures)

Penalty Method

path selection



basic approach (classical method)

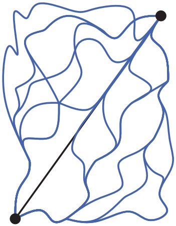
- perform "enough" iterations
  - ⇒ generate set of shortest paths
- select good subset for AG
  (implementation details left open)

when query times matter... (our approach)

- exploit quality measures
  - ⇒ terminate as paths become too long
    (allow maximum stretch w.r.t. original metric)
  - ⇒ select path for AG after each iteration
    (w.r.t. stretch, sharing measures)

Penalty Method

path selection



basic approach (classical method)

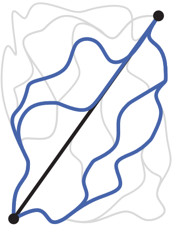
- perform "enough" iterations
  - ⇒ generate set of shortest paths
- select good subset for AG
  (implementation details left open)

when query times matter... (our approach)

- exploit quality measures
  - ⇒ terminate as paths become too long
    (allow maximum stretch w.r.t. original metric)
  - ⇒ select path for AG after each iteration
    (w.r.t. stretch, sharing measures)

Penalty Method

path selection



basic approach (classical method)

- perform "enough" iterations
  - ⇒ generate set of shortest paths
- select good subset for AG
  (implementation details left open)

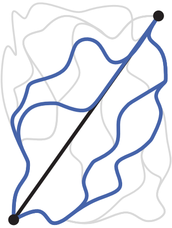
when query times matter... (our approach)

- exploit quality measures
  - ⇒ terminate as paths become too long
    (allow maximum stretch w.r.t. original metric)
  - ⇒ select path for AG after each iteration
    (w.r.t. stretch, sharing measures)

Penalty Method
path selection

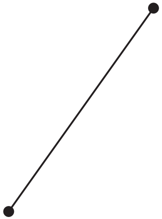
basic approach (classical method)

- perform "enough" iterations
  - ⇒ generate set of shortest paths
- select good subset for AG
  (implementation details left open)



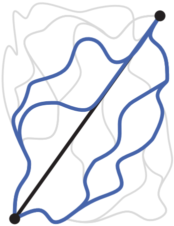
when query times matter... (our approach)

- exploit quality measures
  - ⇒ terminate as paths become too long
    (allow maximum stretch w.r.t. original metric)
  - ⇒ select path for AG after each iteration
    (w.r.t. stretch, sharing measures)



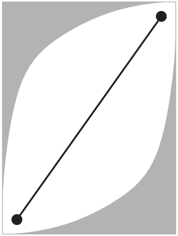
Penalty Method

path selection



basic approach (classical method)

- perform "enough" iterations
  - ⇒ generate set of shortest paths
- select good subset for AG
  (implementation details left open)



when query times matter... (our approach)

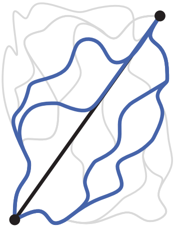
- exploit quality measures
  - ⇒ terminate as paths become too long
    (allow maximum stretch w.r.t. original metric)
  - ⇒ select path for AG after each iteration
    (w.r.t. stretch, sharing measures)

Penalty Method

path selection

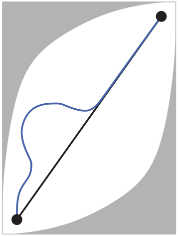
basic approach (classical method)

- perform "enough" iterations
  - ⇒ generate set of shortest paths
- select good subset for AG
  (implementation details left open)



when query times matter... (our approach)

- exploit quality measures
  - ⇒ terminate as paths become too long
    (allow maximum stretch w.r.t. original metric)
  - ⇒ select path for AG after each iteration
    (w.r.t. stretch, sharing measures)

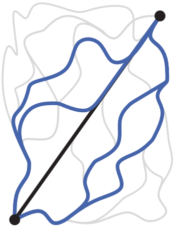


Penalty Method

path selection

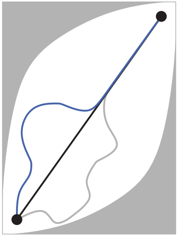
basic approach (classical method)

- perform "enough" iterations
  - ⇒ generate set of shortest paths
- select good subset for AG
  (implementation details left open)



when query times matter... (our approach)

- exploit quality measures
  - ⇒ terminate as paths become too long
    (allow maximum stretch w.r.t. original metric)
  - ⇒ select path for AG after each iteration
    (w.r.t. stretch, sharing measures)

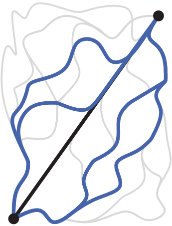


Penalty Method

path selection

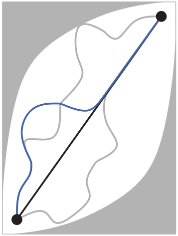
basic approach (classical method)

- perform "enough" iterations
  - ⇒ generate set of shortest paths
- select good subset for AG
  (implementation details left open)



when query times matter... (our approach)

- exploit quality measures
  - ⇒ terminate as paths become too long
    (allow maximum stretch w.r.t. original metric)
  - ⇒ select path for AG after each iteration
    (w.r.t. stretch, sharing measures)

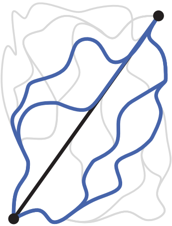


Penalty Method

path selection

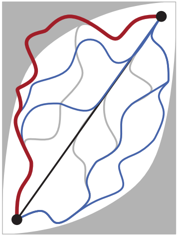
basic approach (classical method)

- perform "enough" iterations
  - ⇒ generate set of shortest paths
- select good subset for AG
  (implementation details left open)



when query times matter... (our approach)

- exploit quality measures
  - ⇒ terminate as paths become too long
    (allow maximum stretch w.r.t. original metric)
  - ⇒ select path for AG after each iteration
    (w.r.t. stretch, sharing measures)

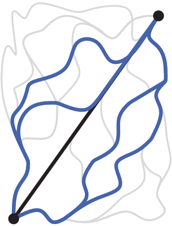


Penalty Method

path selection

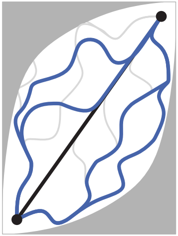
basic approach (classical method)

- perform "enough" iterations
  - ⇒ generate set of shortest paths
- select good subset for AG
  (implementation details left open)



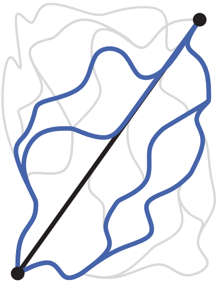
when query times matter... (our approach)

- exploit quality measures
  - ⇒ terminate as paths become too long
    (allow maximum stretch w.r.t. original metric)
  - ⇒ select path for AG after each iteration
    (w.r.t. stretch, sharing measures)



Penalty Method

computing shortest paths (1)



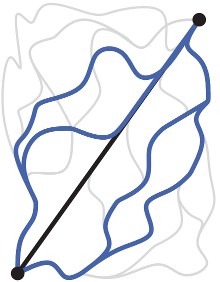
challenges

- compute many shortest paths
  - ⇒ Dijkstra's algorithm [DIJKSTRA 59]
    → used by previous work
    (takes seconds on random queries)
  - ⇒ static speed-up techniques
    (AF, CH, HL, . . . → costly preprocessing)

- arc costs change
  - ⇒ Customizable Route Planning (CRP) [DELLING ET AL. 13]
    → used by our approach
    (takes milliseconds on random queries)

Penalty Method

computing shortest paths (1)



challenges

- compute many shortest paths
  - ⇒ Dijkstra's algorithm [DIJKSTRA 59]
    → used by previous work
    (takes seconds on random queries)
  - ⇒ static speed-up techniques
    (AF, CH, HL, . . . → costly preprocessing)

- arc costs change
  - ⇒ Customizable Route Planning (CRP) [DELLING ET AL. 13]
    → used by our approach
    (takes milliseconds on random queries)
  - ⇒ but still requires some kind of preprocessing

Penalty Method

computing shortest paths (2)

CRP preprocessing

- structural preprocessing
  - takes minutes to hours
    (required once, can be done offline)
  - multi-level partitioning
  - adding shortcut arcs
    (boundary nodes of each cell become cliques)

- metric customization
  - compute shortcut costs
    (required when arc costs change)
  - takes seconds[1] / tenths of a second[2]
    (using multiple cores and vector units)

Penalty Method

[DELLING ET AL. 13][1]
[DELLING&WERNECK 13][2]

computing shortest paths (2)

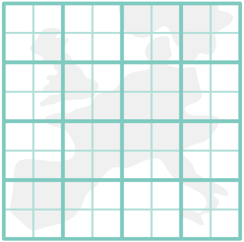
CRP preprocessing

- structural preprocessing
  - takes minutes to hours
    (required once, can be done offline)
  - multi-level partitioning
  - adding shortcut arcs
    (boundary nodes of each cell become cliques)

- metric customization
  - compute shortcut costs
    (required when arc costs change)
  - takes seconds[1] / tenths of a second[2]
    (using multiple cores and vector units)

Penalty Method

computing shortest paths (2)



CRP preprocessing

- structural preprocessing
  - takes minutes to hours
    (required once, can be done offline)
  - multi-level partitioning
  - adding shortcut arcs
    (boundary nodes of each cell become cliques)

- metric customization
  - compute shortcut costs
    (required when arc costs change)
  - takes seconds[1] / tenths of a second[2]
    (using multiple cores and vector units)

Penalty Method

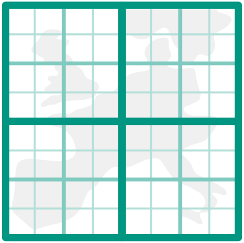
computing shortest paths (2)

[Delling et al. 13][1]
[Delling&Werneck 13][2]

Karlsruhe Institute of Technology

CRP preprocessing

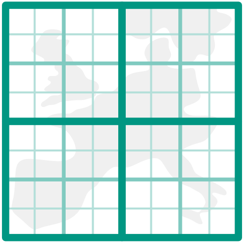
- structural preprocessing
  - takes minutes to hours
    (required once, can be done offline)
  - multi-level partitioning
  - adding shortcut arcs
    (boundary nodes of each cell become cliques)



- metric customization
  - compute shortcut costs
    (required when arc costs change)
  - takes seconds[1] / tenths of a second[2]
    (using multiple cores and vector units)

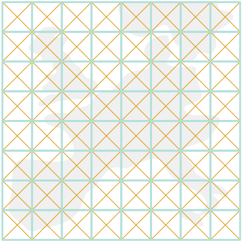
Penalty Method

computing shortest paths (2)



CRP preprocessing

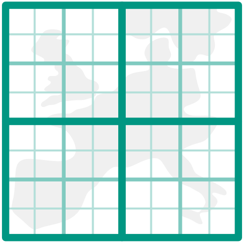
- structural preprocessing
  - takes minutes to hours
    (required once, can be done offline)
  - multi-level partitioning
  - adding shortcut arcs
    (boundary nodes of each cell become cliques)



- metric customization
  - compute shortcut costs
    (required when arc costs change)
  - takes seconds[1] / tenths of a second[2]
    (using multiple cores and vector units)

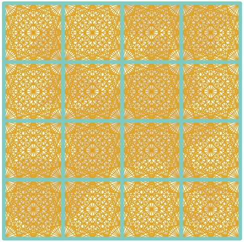
Penalty Method

computing shortest paths (2)



CRP preprocessing

- structural preprocessing
  - takes minutes to hours
    (required once, can be done offline)
  - multi-level partitioning
  - adding shortcut arcs
    (boundary nodes of each cell become cliques)



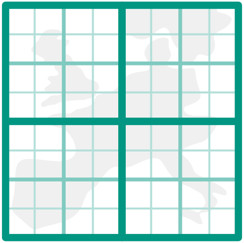
- metric customization
  - compute shortcut costs
    (required when arc costs change)
  - takes seconds[1] / tenths of a second[2]
    (using multiple cores and vector units)

Penalty Method
computing shortest paths (2)

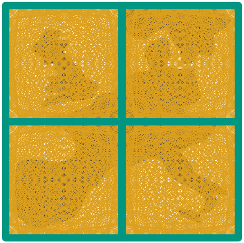
[Delling et al. 13][1]
[Delling&Werneck 13][2]

CRP preprocessing

- structural preprocessing
  - takes minutes to hours
    (required once, can be done offline)
  - multi-level partitioning
  - adding shortcut arcs
    (boundary nodes of each cell become cliques)

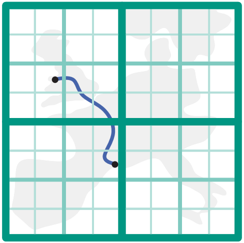


- metric customization
  - compute shortcut costs
    (required when arc costs change)
  - takes seconds[1] / tenths of a second[2]
    (using multiple cores and vector units)



Penalty Method

computing shortest paths (3)



adaptive customization

- only update cells with changes
- only update *k* levels

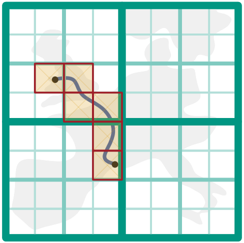
  ⇒ restrict query algorithm to *k* levels
  ⇒ preprocessing times ⟺ query times
  ⇒ beneficial for short queries (less overhead)

- dynamic level selection

  (depending on hop count, optimized on different query set)

Penalty Method

computing shortest paths (3)



adaptive customization

- only update cells with changes
- only update k levels
  - ⇒ restrict query algorithm to k levels
  - ⇒ preprocessing times ⟺ query times
  - ⇒ beneficial for short queries (less overhead)
- dynamic level selection
  (depending on hop count, optimized on different query set)

Penalty Method

computing shortest paths (3)

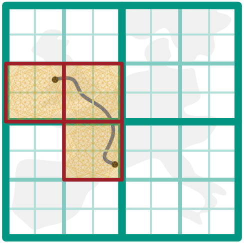
adaptive customization

- only update cells with changes
- only update *k* levels

  ⇒ restrict query algorithm to *k* levels
  ⇒ preprocessing times ⟺ query times
  ⇒ beneficial for short queries (less overhead)

- dynamic level selection

  (depending on hop count, optimized on different query set)



Penalty Method

computing shortest paths (3)

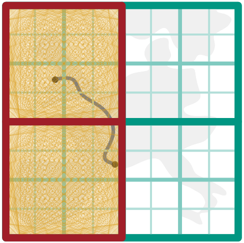
adaptive customization

- only update cells with changes
- only update *k* levels

  $\Rightarrow$ restrict query algorithm to *k* levels
  $\Rightarrow$ preprocessing times $\Longleftrightarrow$ query times
  $\Rightarrow$ beneficial for short queries (less overhead)

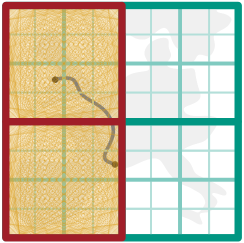
- dynamic level selection

  (depending on hop count, optimized on different query set)



Penalty Method

computing shortest paths (3)



adaptive customization

- only update cells with changes
- only update k levels

  $\Rightarrow$ restrict query algorithm to k levels
  $\Rightarrow$ preprocessing times $\Longleftrightarrow$ query times
  $\Rightarrow$ beneficial for short queries (less overhead)

- dynamic level selection
  (depending on hop count, optimized on different query set)

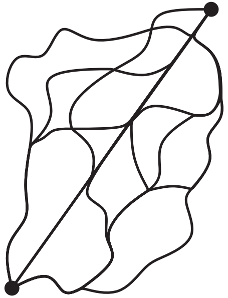
$\Rightarrow$ tens of milliseconds for query and customization

Penalty Method

alternative route extraction (1)

how to extract alternative routes?

- take constituting routes?
  $\Rightarrow$ misses synergy effects
- take any route?
  $\Rightarrow$ arbitrarily bad
- our approach: *CRP-π*
  - two-step procedure
    (via-node & penalty methods)
  - extraction on small graph
    (expensive methods become feasible)

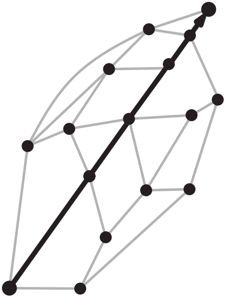


Penalty Method

alternative route extraction (1)

how to extract alternative routes?

- take constituting routes?
  ⇒ misses synergy effects
- take any route?
  ⇒ arbitrarily bad
- our approach: *CRP-π*
  - two-step procedure
    (via-node & penalty methods)
  - extraction on small graph
    (expensive methods become feasible)

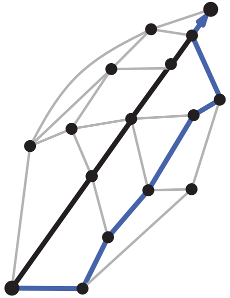


Penalty Method

alternative route extraction (1)

how to extract alternative routes?

- take constituting routes?
  ⇒ misses synergy effects
- take any route?
  ⇒ arbitrarily bad
- our approach: *CRP-π*
  - two-step procedure
    (via-node & penalty methods)
  - extraction on small graph
    (expensive methods become feasible)

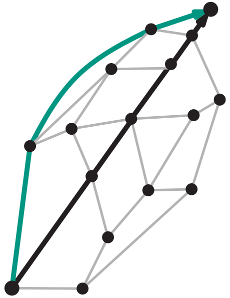


Penalty Method

alternative route extraction (1)

how to extract alternative routes?

- take constituting routes?
  ⇒ misses synergy effects
- take any route?
  ⇒ arbitrarily bad
- our approach: *CRP-π*
  - two-step procedure
    (via-node & penalty methods)
  - extraction on small graph
    (expensive methods become feasible)



Penalty Method

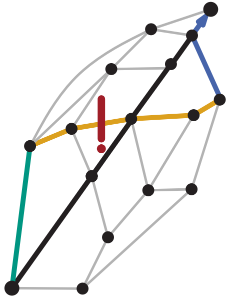
alternative route extraction (1)

how to extract alternative routes?

- take constituting routes?
  $\Rightarrow$ misses synergy effects

- take any route?
  $\Rightarrow$ arbitrarily bad

- our approach: *CRP-π*
  - two-step procedure
    (via-node & penalty methods)
  - extraction on small graph
    (expensive methods become feasible)

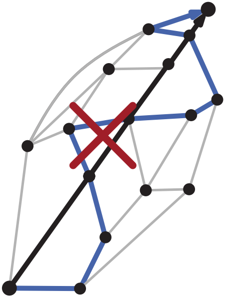


Penalty Method

alternative route extraction (1)

how to extract alternative routes?

- take constituting routes?
  ⇒ misses synergy effects
- take any route?
  ⇒ arbitrarily bad
- our approach: *CRP-π*
  - two-step procedure
    (via-node & penalty methods)
  - extraction on small graph
    (expensive methods become feasible)

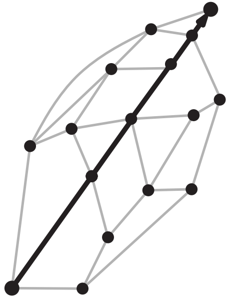


Penalty Method

alternative route extraction (2)

procedure

1. via-node approach
   - based on X-BDV
     → local optimality disabled (not applicable in AG)
   - exhaustively compute routes
   - keep all routes



2. penalty method
   - based on classical penalty method
     → initialize with X-BDV routes
   - fixed number of iterations (stretch no good termination criterion)
   - keep good routes (w.r.t. quality measures)

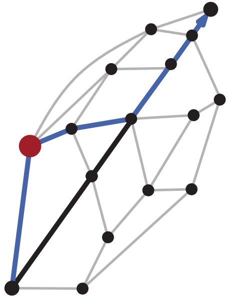
Penalty Method

alternative route extraction (2)

Karlsruhe Institute of Technology

procedure

1. via-node approach
   - based on X-BDV
     → local optimality disabled (not applicable in AG)
   - exhaustively compute routes
   - keep all routes



2. penalty method
   - based on classical penalty method
     → initialize with X-BDV routes
   - fixed number of iterations (stretch no good termination criterion)
   - keep good routes (w.r.t. quality measures)

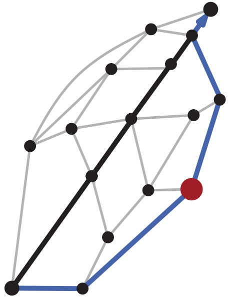
Penalty Method

alternative route extraction (2)

Karlsruhe Institute of Technology

procedure

1. via-node approach
   - based on X-BDV
     $\rightarrow$ local optimality disabled (not applicable in AG)
   - exhaustively compute routes
   - keep all routes

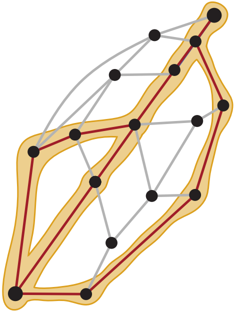


2. penalty method
   - based on classical penalty method
     $\rightarrow$ initialize with X-BDV routes
   - fixed number of iterations (stretch no good termination criterion)
   - keep good routes (w.r.t. quality measures)

Penalty Method
alternative route extraction (2)

procedure

1. via-node approach
   - based on X-BDV
     → local optimality disabled (not applicable in AG)
   - exhaustively compute routes
   - keep all routes



2. penalty method
   - based on classical penalty method
     → initialize with X-BDV routes
   - fixed number of iterations (stretch no good termination criterion)
   - keep good routes (w.r.t. quality measures)

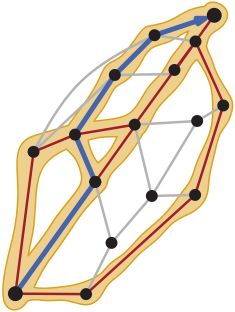
Penalty Method

alternative route extraction (2)

procedure

1. via-node approach
   - based on X-BDV
     → local optimality disabled (not applicable in AG)
   - exhaustively compute routes
   - keep all routes

2. penalty method
   - based on classical penalty method
     → initialize with X-BDV routes
   - fixed number of iterations (stretch no good termination criterion)
   - keep good routes (w.r.t. quality measures)



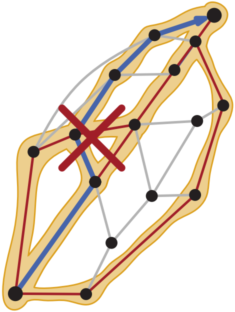
Penalty Method

alternative route extraction (2)

procedure

1. via-node approach
   - based on X-BDV
     → local optimality disabled (not applicable in AG)
   - exhaustively compute routes
   - keep all routes

2. penalty method
   - based on classical penalty method
     → initialize with X-BDV routes
   - fixed number of iterations (stretch no good termination criterion)
   - keep good routes (w.r.t. quality measures)



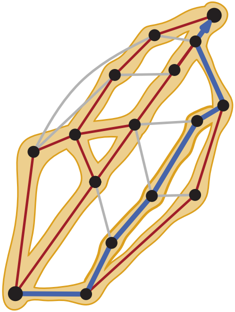
Penalty Method

alternative route extraction (2)

Karlsruhe Institute of Technology

procedure

1. via-node approach
   - based on X-BDV
     → local optimality disabled (not applicable in AG)
   - exhaustively compute routes
   - keep all routes



2. penalty method
   - based on classical penalty method
     → initialize with X-BDV routes
   - fixed number of iterations (stretch no good termination criterion)
   - keep good routes (w.r.t. quality measures)

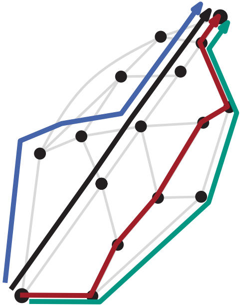
Penalty Method

alternative route extraction (2)

procedure

1. via-node approach
   - based on X-BDV
     → local optimality disabled (not applicable in AG)
   - exhaustively compute routes
   - keep all routes

2. penalty method
   - based on classical penalty method
     → initialize with X-BDV routes
   - fixed number of iterations (stretch no good termination criterion)
   - keep good routes (w.r.t. quality measures)



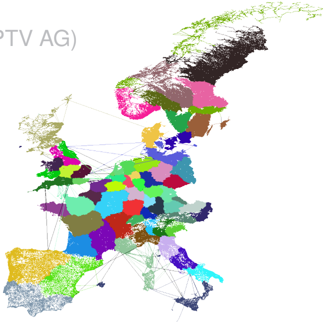
Experimental Setup

hardware/software

- 4 Intel Xeon E5-4640 @ 2.4 GHz, 512 GiB RAM (32 cores, total)
- Ubuntu 12.04, gcc 4.6.1 (full optimizations)

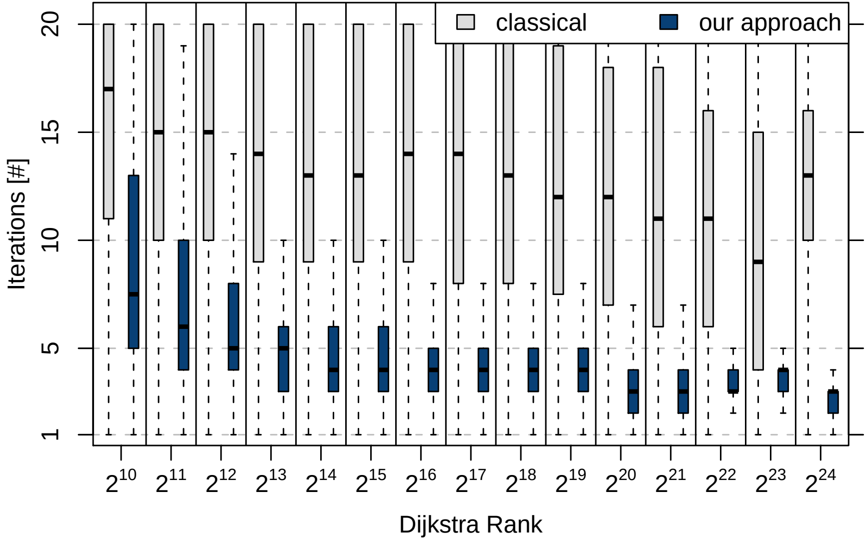
data

- road network of Western Europe (provided by PTV AG)
  - directed, weighted graph, single SCC
  - 18 million vertices, 24 million arcs
    (degree two vertices removed)
  - travel-time metric
- 1 000 queries at random / data point
  (of random Dijkstra rank / of fixed Dijkstra rank)



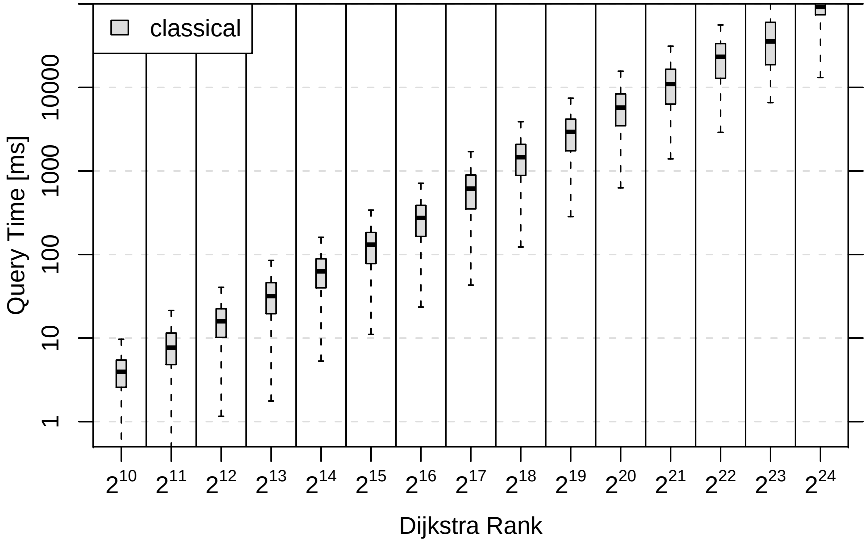
Runtime Analysis

iterations of penalty method



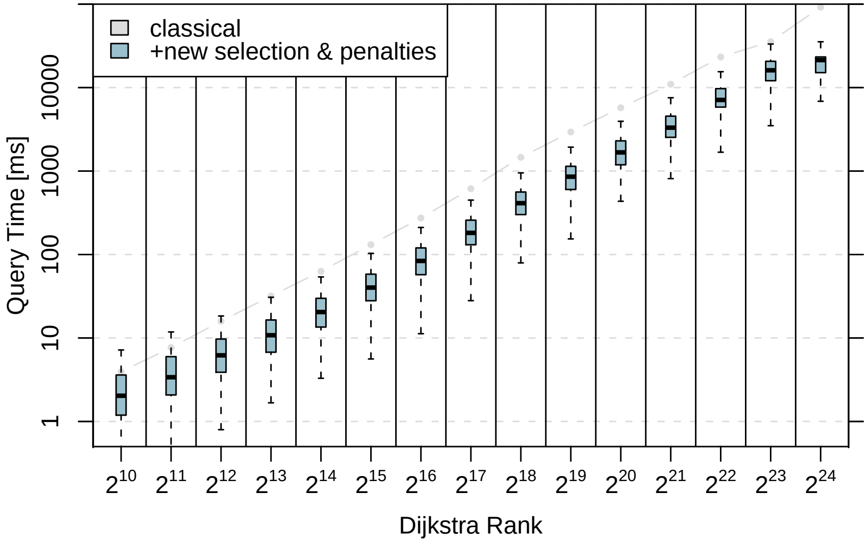
Runtime Analysis

engineering impact



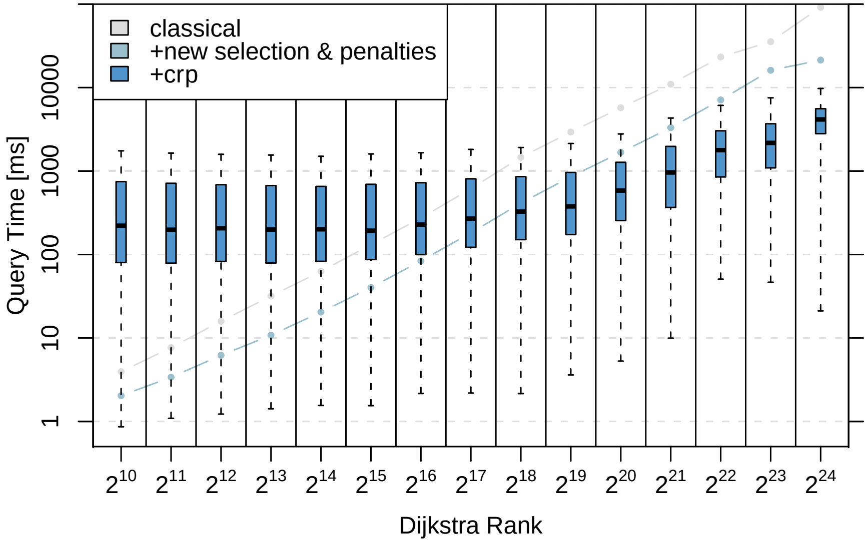
Runtime Analysis

engineering impact



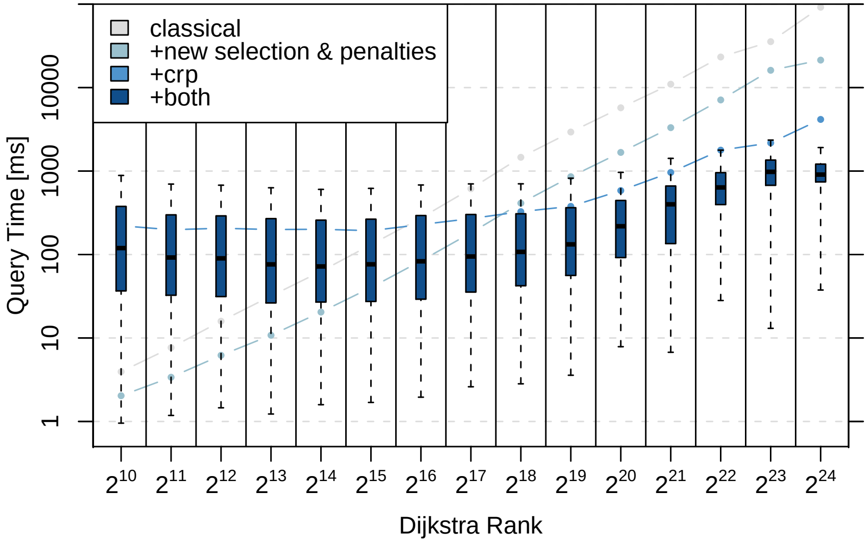
Runtime Analysis

engineering impact



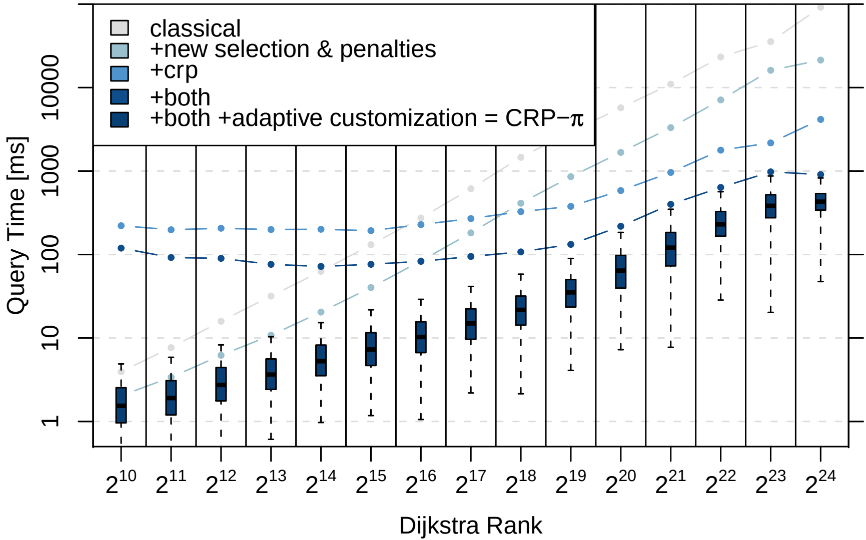
Runtime Analysis

engineering impact



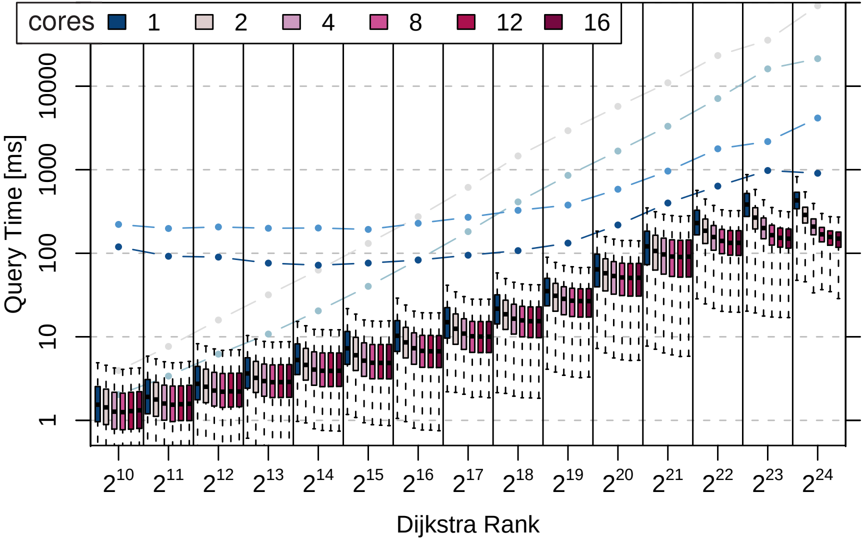
Runtime Analysis

engineering impact



Runtime Analysis

engineering impact – multi-cores



Quality Analysis

alternative graph quality

algorithm	rating	queries	decision arcs
[BADER ET AL. 11]	3.21	100	$\leq$ 10.0
[RADERMACHER 12]	2.89	1 000	7.0
CRP-π	3.32	1 000	(unfiltered) 17.4
CRP-π	2.89	1 000	(filtered) 9.5

- comparable results to previous penalty methods
  - best previous work only considered tiny test set
- filtering to reduce decision arcs ($\approx$100μs)
  - only for comparison to previous penalty methods
  - reduces potential for extracting multiple alternative routes (not applied in subsequent alternative route analysis)

Quality Analysis

alternative route quality

#	algorithm	success [%]	stretch [%]	sharing [%]	optimality [%]
1st	X-BDV	96.0	10.0	41.8	75.4
	X-CHV	89.6	80.4	40.6	68.1
	CRP-π	96.3	42.9	31.9	26.9
2nd	X-BDV	87.6	13.8	59.5	65.1
	X-CHV	72.5	269.0	57.6	57.2
	CRP-π	84.0	47.6	45.9	22.1
3rd	X-BDV	75.5	17.2	65.6	54.6
	X-CHV	51.4	214.0	63.6	46.8
	CRP-π	62.9	67.4	51.8	15.9

- comparable success rates (corresponds to runtimes)
- reasonable path quality measures (lower stretch $\Longleftrightarrow$ higher sharing)
- limited similarity to via-node alternativs (77.9% | 72.7% | 65.5%)

Quality Analysis

alternative route quality

#	algorithm	success [%]	stretch [%]	sharing [%]	optimality [%]
1st	X-BDV	96.0	10.0	41.8	75.4
	X-CHV	89.6	80.4	40.6	68.1
	CRP-π	96.3	42.9	31.9	26.9
2nd	X-BDV	87.6	13.8	59.5	65.1
	X-CHV	72.5	269.0	57.6	57.2
	CRP-π	84.0	47.6	45.9	22.1
3rd	X-BDV	75.5	17.2	65.6	54.6
	X-CHV	51.4	214.0	63.6	46.8
	CRP-π	62.9	67.4	51.8	15.9

- comparable success rates (corresponds to runtimes)
- reasonable path quality measures (lower stretch $\Longleftrightarrow$ higher sharing)
- limited similarity to via-node alternativs (77.9% | 72.7% | 65.5%)

Quality Analysis

alternative route quality

#	algorithm	success [%]	stretch [%]	sharing [%]	optimality [%]
1st	X-BDV	96.0	10.0	41.8	75.4
	X-CHV	89.6	80.4	40.6	68.1
	CRP-π	96.3	42.9	31.9	26.9
2nd	X-BDV	87.6	13.8	59.5	65.1
	X-CHV	72.5	269.0	57.6	57.2
	CRP-π	84.0	47.6	45.9	22.1
3rd	X-BDV	75.5	17.2	65.6	54.6
	X-CHV	51.4	214.0	63.6	46.8
	CRP-π	62.9	67.4	51.8	15.9

- comparable success rates (corresponds to runtimes)
- reasonable path quality measures (lower stretch $\Longleftrightarrow$ higher sharing)
- limited similarity to via-node alternativs (77.9% | 72.7% | 65.5%)

Quality Analysis

alternative route quality

#	algorithm	success [%]	stretch [%]	sharing [%]	optimality [%]
1st	X-BDV	96.0	10.0	41.8	75.4
	X-CHV	89.6	80.4	40.6	68.1
	CRP-π	96.3	42.9	31.9	26.9
2nd	X-BDV	87.6	13.8	59.5	65.1
	X-CHV	72.5	269.0	57.6	57.2
	CRP-π	84.0	47.6	45.9	22.1
3rd	X-BDV	75.5	17.2	65.6	54.6
	X-CHV	51.4	214.0	63.6	46.8
	CRP-π	62.9	67.4	51.8	15.9

- comparable success rates (corresponds to runtimes)
- reasonable path quality measures (lower stretch $\Longleftrightarrow$ higher sharing)
- limited similarity to via-node alternativs (77.9% | 72.7% | 65.5%)

Conclusion

Summary

- first fast implemenation of penalty method, suited for interactive use
  (utilizing vector units and multi-core capabilities of modern CPUs)
- extracted routes of high quality, and distinct from via-node approach
  (first quantitative analysis of extracted routes from alternative graphs)

Open Problems

- general classification scheme for good alternatives
  (set of criteria, not tailored to a specific approach)
- improve runtime to compete with via-node approach
  (combination with [PARASKEVOPOULOS&ZAROLIAGIS 13] seems promising)

Thank you for your attention!

Time for questions!

References

[DIJKSTRA 59]
A Note on Two Problems in Connexion with Graphs

[ABRAHAM ET AL. 10]
Alternative Routes in Road Networks

[BADER ET AL. 11]
Alternative Route Graphs in Road Networks

[DELLING ET AL. 13]
Customizable Route Planning

[DELLING&WERNECK 13]
Faster Customization of Road Networks

[PARASKEVOPOULOS&ZAROLIAGIS 13]
Improved Alternative Route Planning

[RADERMACHER 12]
Schnelle Berechnung von Alternativgraphen

Appendix

Karlsruhe Institute of Technology

backup slides

Motivation

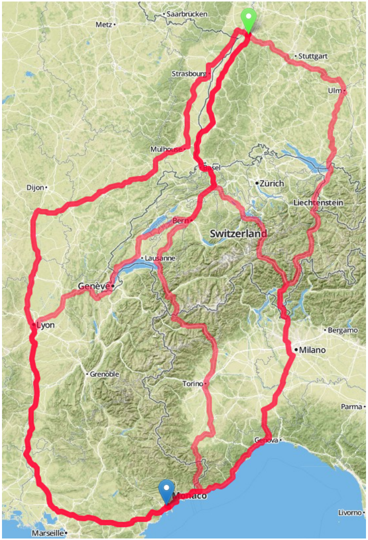
why consider alternative routes?

business perspective

- provide options
  (users have varied preferences)
- overcome flaws in model and data
  (shortest paths need not be best in reality)

research perspective

- building blocks
  (traffic simulation, stochastic routing)
- hard optimization problems
  (quality guarantees)



Motivation

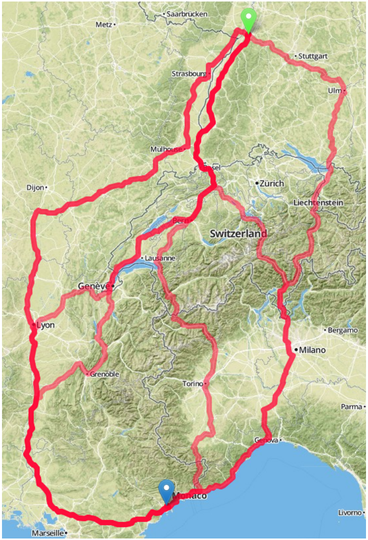
why consider alternative routes?

business perspective

- provide options
  (users have varied preferences)
- overcome flaws in model and data
  (shortest paths need not be best in reality)

research perspective

- building blocks
  (traffic simulation, stochastic routing)
- hard optimization problems
  (quality guarantees)



Motivation

why consider alternative routes?

business perspective

- provide options
  (users have varied preferences)
- overcome flaws in model and data
  (shortest paths need not be best in reality)

research perspective

- building blocks
  (traffic simulation, stochastic routing)
- hard optimization problems
  (quality guarantees)

Motivation

why consider alternative routes?

business perspective

- provide options
  (users have varied preferences)
- overcome flaws in model and data
  (shortest paths need not be best in reality)

research perspective

- building blocks
  (traffic simulation, stochastic routing)
- hard optimization problems
  (quality guarantees)

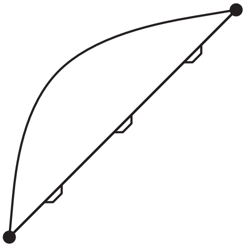
Related Work

peering into the distant past...

Karlsruhe Institute of Technology

simple approaches to find distinct routes

- *k*-shortest path
  (meaningful alternatives only for large *k*)
- multi-criteria optimization
  (distance $\Longleftrightarrow$ difference)
- time-dependent routes
  (alternatives not guaranteed, limited data)



Related Work

peering into the distant past...

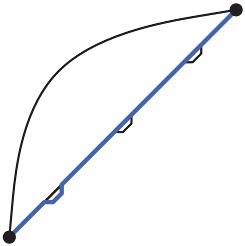
Karlsruhe Institute of Technology

simple approaches to find distinct routes

- *k*-shortest path
  (meaningful alternatives only for large *k*)

- multi-criteria optimization
  (distance ⟺ difference)

- time-dependent routes
  (alternatives not guaranteed, limited data)



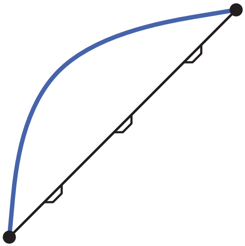
Related Work

peering into the distant past...

Karlsruhe Institute of Technology

simple approaches to find distinct routes

- *k*-shortest path
  (meaningful alternatives only for large *k*)
- multi-criteria optimization
  (distance ⟺ difference)
- time-dependent routes
  (alternatives not guaranteed, limited data)

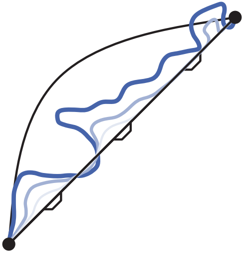


Related Work

peering into the distant past...

simple approaches to find distinct routes

- *k*-shortest path
  (meaningful alternatives only for large *k*)
- multi-criteria optimization
  (distance ⟺ difference)
- time-dependent routes
  (alternatives not guaranteed, limited data)



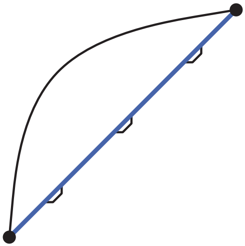
Related Work

peering into the distant past...

Karlsruhe Institute of Technology

simple approaches to find distinct routes

- *k*-shortest path
  (meaningful alternatives only for large *k*)
- multi-criteria optimization
  (distance ⟺ difference)
- time-dependent routes
  (alternatives not guaranteed, limited data)



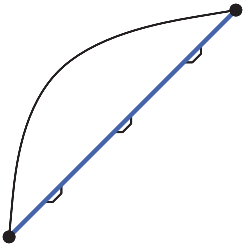
Related Work

peering into the distant past...

Karlsruhe Institute of Technology

simple approaches to find distinct routes

- *k*-shortest path
  (meaningful alternatives only for large *k*)
- multi-criteria optimization
  (distance $\Longleftrightarrow$ difference)
- time-dependent routes
  (alternatives not guaranteed, limited data)



insufficient solutions to the problem

Penalty Method

penalization (1)

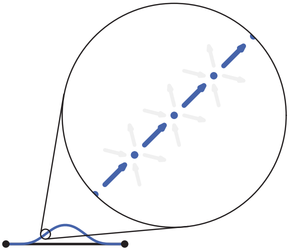
requirements

- enable discovery of diverse routes
  - ⇒ penalties on arcs of current route
    (discourage previous routes)
  - ⇒ penalties on adjoined arcs
    (discourage meandering)

- quick discovery of diverse routes
  - ⇒ generate lots of new information in each iteration
  - ⇒ speed $\Longleftrightarrow$ quality

Penalty Method

penalization (1)

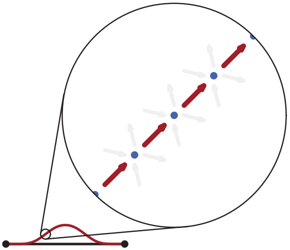


requirements

- enable discovery of diverse routes
  - ⇒ penalties on arcs of current route
    (discourage previous routes)
  - ⇒ penalties on adjoined arcs
    (discourage meandering)
- quick discovery of diverse routes
  - ⇒ generate lots of new information in each iteration
  - ⇒ speed ⟺ quality

Penalty Method

penalization (1)



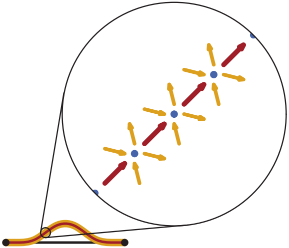
requirements

- enable discovery of diverse routes
  - ⇒ penalties on arcs of current route
    (discourage previous routes)
  - ⇒ penalties on adjoined arcs
    (discourage meandering)

- quick discovery of diverse routes
  - ⇒ generate lots of new information in each iteration
  - ⇒ speed $\Longleftrightarrow$ quality

Penalty Method

penalization (1)



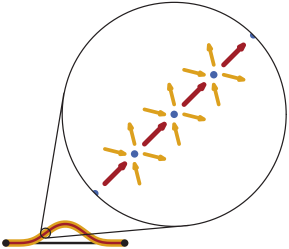
requirements

- enable discovery of diverse routes
  - ⇒ penalties on arcs of current route
    (discourage previous routes)
  - ⇒ penalties on adjoined arcs
    (discourage meandering)

- quick discovery of diverse routes
  - ⇒ generate lots of new information in each iteration
  - ⇒ speed $\Longleftrightarrow$ quality

Penalty Method

penalization (1)



requirements

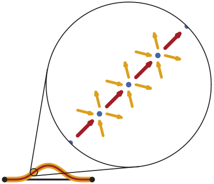
- enable discovery of diverse routes
  - ⇒ penalties on arcs of current route
    (discourage previous routes)
  - ⇒ penalties on adjoined arcs
    (discourage meandering)

- quick discovery of diverse routes
  - ⇒ generate lots of new information in each iteration
  - ⇒ speed ⟺ quality

⇒ how to choose penalty values?

Penalty Method

penalization (2)



additive penalties (classical method)

- add fraction of arc/path costs (w.r.t. original metric)
  $\Rightarrow$ shortest paths only change slowly

geometrically growing penalties (our approach)

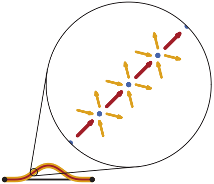
- multiply arc costs on current route by small factor
  $\Rightarrow$ often used paths quickly become undesirable
- add $\frac{1}{2}\sqrt{\text{(current route cost)}}$ to adjoined arc costs
  $\Rightarrow$ discourages short detours on long routes

Penalty Method

penalization (2)

additive penalties (classical method)

- add fraction of arc/path costs (w.r.t. original metric)
  $\Rightarrow$ shortest paths only change slowly

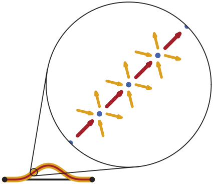


geometrically growing penalties (our approach)

- multiply arc costs on current route by small factor
  $\Rightarrow$ often used paths quickly become undesirable
- add $\frac{1}{2}\sqrt{(\text{current route cost})}$ to adjoined arc costs
  $\Rightarrow$ discourages short detours on long routes

Penalty Method

penalization (2)

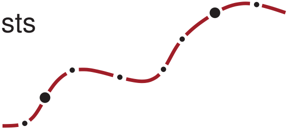


additive penalties (classical method)

- add fraction of arc/path costs (w.r.t. original metric)
  $\Rightarrow$ shortest paths only change slowly

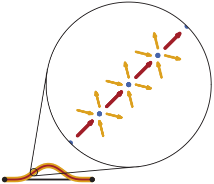
geometrically growing penalties (our approach)

- multiply arc costs on current route by small factor
  $\Rightarrow$ often used paths quickly become undesirable
- add $\frac{1}{2}\sqrt{(\text{current route cost})}$ to adjoined arc costs
  $\Rightarrow$ discourages short detours on long routes



Penalty Method

penalization (2)



additive penalties (classical method)

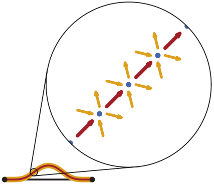
- add fraction of arc/path costs (w.r.t. original metric)
  $\Rightarrow$ shortest paths only change slowly

geometrically growing penalties (our approach)

- multiply arc costs on current route by small factor
  $\Rightarrow$ often used paths quickly become undesirable
- add $\frac{1}{2}\sqrt{(\text{current route cost})}$ to adjoined arc costs
  $\Rightarrow$ discourages short detours on long routes

Penalty Method

penalization (2)

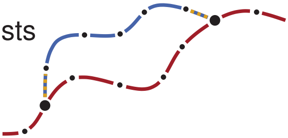


additive penalties (classical method)

- add fraction of arc/path costs (w.r.t. original metric)
  $\Rightarrow$ shortest paths only change slowly

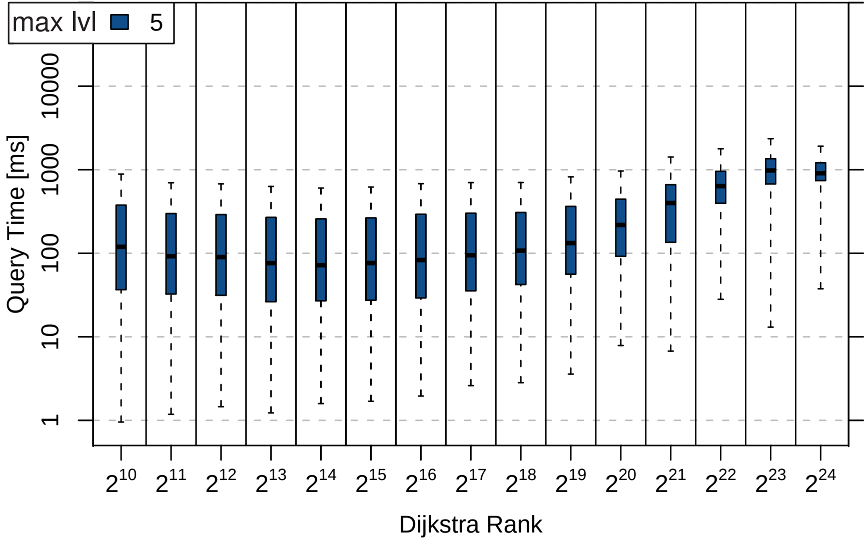
geometrically growing penalties (our approach)

- multiply arc costs on current route by small factor
  $\Rightarrow$ often used paths quickly become undesirable
- add $\frac{1}{2}\sqrt{(\text{current route cost})}$ to adjoined arc costs
  $\Rightarrow$ discourages short detours on long routes



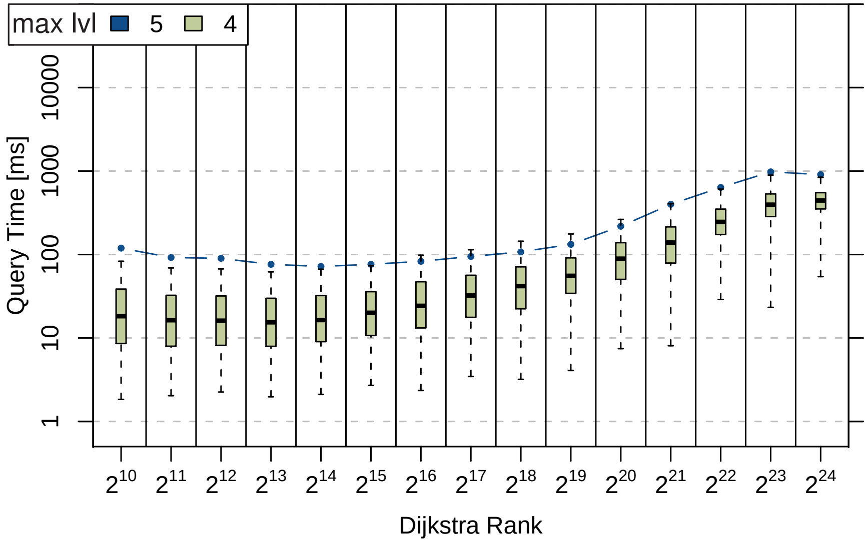
Runtime Analysis

restriction of maximum crp level



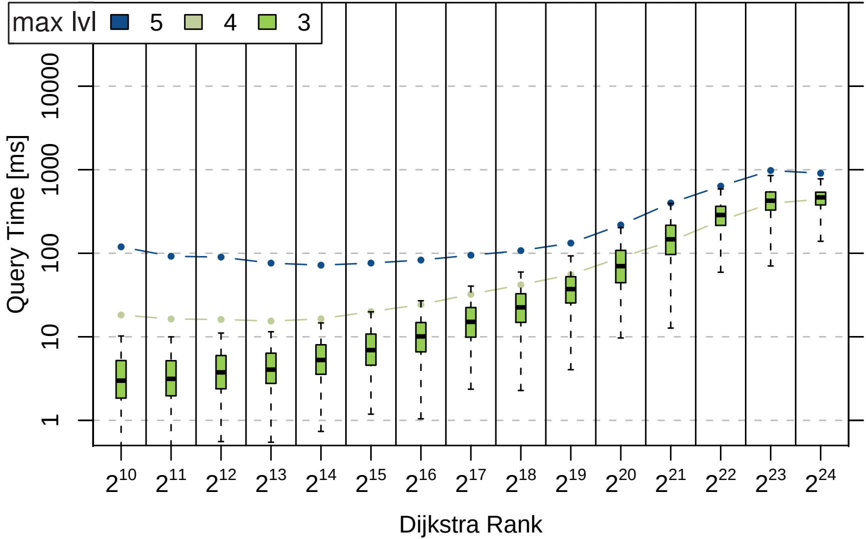
Runtime Analysis

restriction of maximum crp level



Runtime Analysis

restriction of maximum crp level

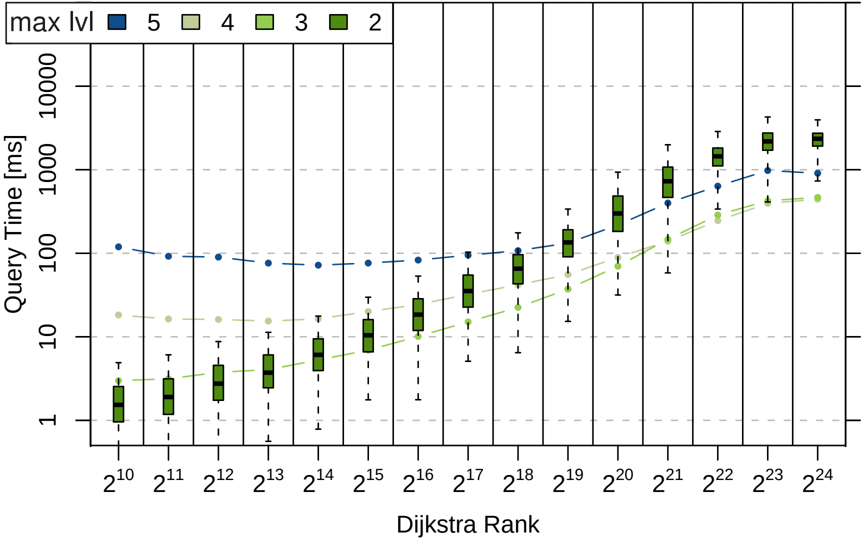


Kobitzsch, Radermacher, Schieferdecker:
Evolution and Evaluation of the Penalty Method for Alternative Graphs

Institute of Theoretical Informatics
Algorithmics

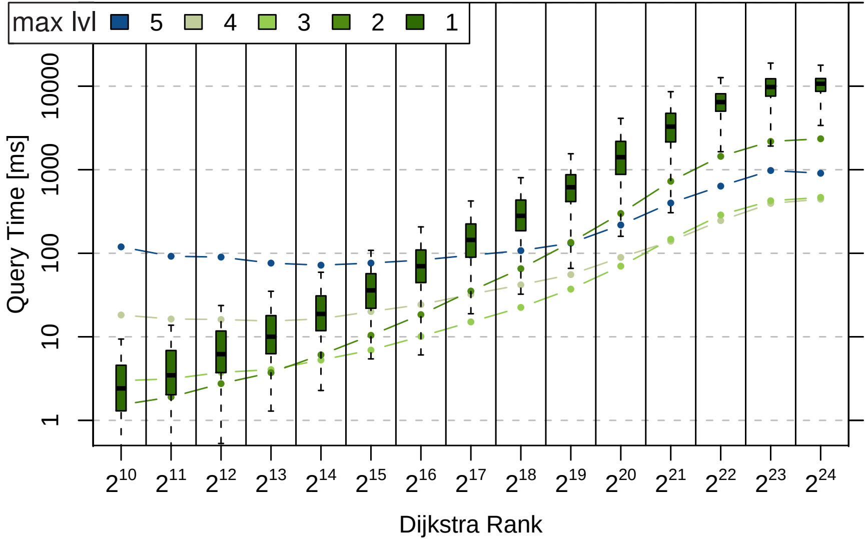
Runtime Analysis

restriction of maximum crp level



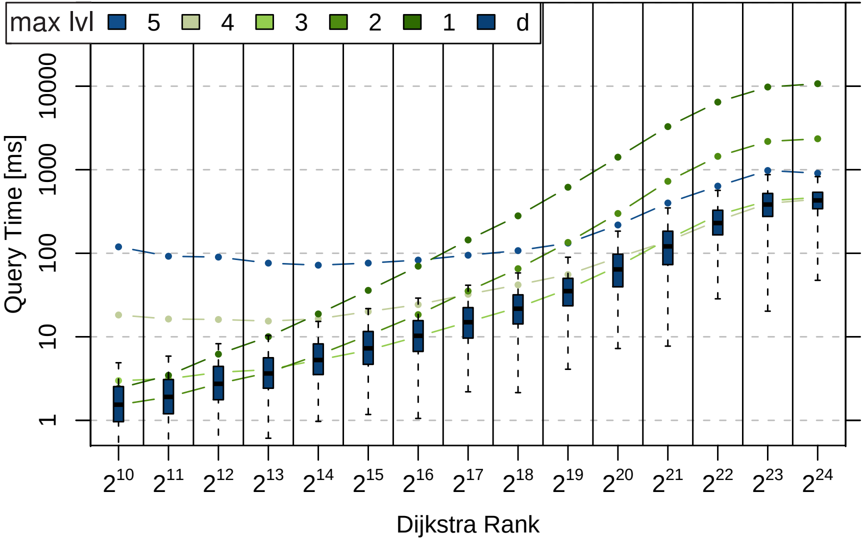
Runtime Analysis

restriction of maximum crp level



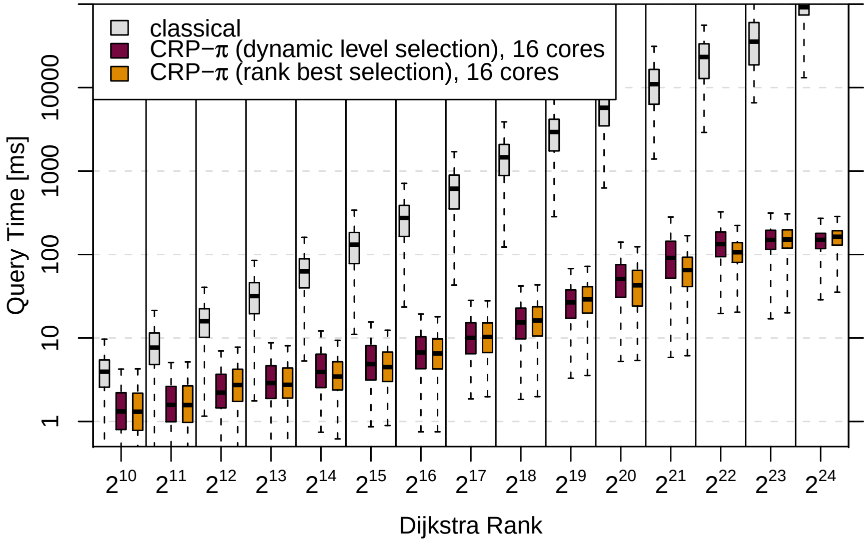
Runtime Analysis

restriction of maximum crp level



Runtime Analysis

optimal selection of maximum crp level



Quality Analysis

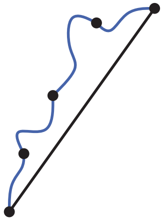
alternative route structure

differences between penalty & via-node alternatives

- procedure
  - consider all CRP-π routes
  - for each, find via-node alternative with most overlap
    (test each vertex on CRP-π route as via-node)
- average maximum overlap
  - 77.9% | 72.7% | 65.5% (1st – 3rd alternative)
  - higher order alternatives increasingly distinct
    (first routes likely extracted by via-node approach)

Quality Analysis

alternative route structure

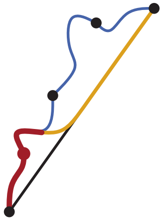


differences between penalty & via-node alternatives

- procedure
  - consider all CRP-π routes
  - for each, find via-node alternative with most overlap
    (test each vertex on CRP-π route as via-node)
- average maximum overlap
  - 77.9% | 72.7% | 65.5% (1st – 3rd alternative)
  - higher order alternatives increasingly distinct
    (first routes likely extracted by via-node approach)

Quality Analysis

alternative route structure



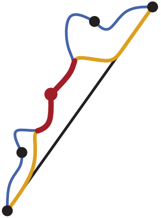
differences between penalty & via-node alternatives

- procedure
  - consider all CRP-π routes
  - for each, find via-node alternative with most overlap
    (test each vertex on CRP-π route as via-node)

- average maximum overlap
  - 77.9% | 72.7% | 65.5% (1st – 3rd alternative)
  - higher order alternatives increasingly distinct
    (first routes likely extracted by via-node approach)

Quality Analysis

alternative route structure



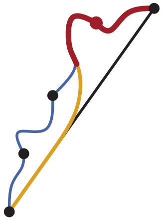
differences between penalty & via-node alternatives

- procedure
  - consider all CRP-π routes
  - for each, find via-node alternative with most overlap
    (test each vertex on CRP-π route as via-node)

- average maximum overlap
  - 77.9% | 72.7% | 65.5% (1st – 3rd alternative)
  - higher order alternatives increasingly distinct
    (first routes likely extracted by via-node approach)

Quality Analysis

alternative route structure



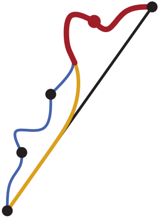
differences between penalty & via-node alternatives

- procedure
  - consider all CRP-π routes
  - for each, find via-node alternative with most overlap
    (test each vertex on CRP-π route as via-node)

- average maximum overlap
  - 77.9% | 72.7% | 65.5% (1st – 3rd alternative)
  - higher order alternatives increasingly distinct
    (first routes likely extracted by via-node approach)

Quality Analysis

alternative route structure



differences between penalty & via-node alternatives

- procedure
  - consider all CRP-π routes
  - for each, find via-node alternative with most overlap
    (test each vertex on CRP-π route as via-node)
- average maximum overlap
  - 77.9% | 72.7% | 65.5% (1st – 3rd alternative)
  - higher order alternatives increasingly distinct
    (first routes likely extracted by via-node approach)

$\Rightarrow$ CRP-π provides distinct routes a via-node approach cannot find