Alternative Routes and Route Corridors

Moritz Kobitzsch – moritz.kobitzsch@kit.edu
Introduction

Moritz Kobitzsch

- general interest in algorithms, especially shortest path algorithms
- Karlsruhe Institute of Technology
- parts of this talk on ongoing research

Collaborators

- Daniel Delling  
  Microsoft Research, Silicon Valley  
  daniel.delling@microsoft.com

- Renato Werneck  
  Microsoft Research, Silicon Valley  
  renato.werneck@microsoft.com

- Dennis Luxen  
  Karlsruhe Institute of Technology  
  dennis.luxen@kit.edu

- Dennis Schieferdecker  
  Karlsruhe Institute of Technology  
  dennis.schieferdecker@kit.edu
Stochastic On Time Arrival Problem

Definition
Given a graph $G = (V, A)$, $|V| = n$, $|A| = m$, for each $a \in A$ a probability distribution $p : \mathbb{N} \mapsto [0, 1]$, depicting the cumulative probability to traverse an arc in a given time $T \in \mathbb{N}$, as well as a time budget $B$ and a source node $s$ as well as a target node $t$:

Goal: Find the optimal strategy, starting at $s$, maximizing the probability of arriving at $t$ within the budget $B$. 
Strategy

Problems

- strategy depends on so far **experienced** travel times
  - optimal strategy might contain **loops**
  - computation has to look at each node multiple times

- each edge relaxation represents a costly convolution
Strategy

Problems

- strategy depends on so far **experienced** travel times
  - optimal strategy might contain loops
  - computation has to look at each node multiple times
- each edge relaxation represents a costly convolution

Grid-Graph

- 64 × 64 nodes
- travel time identical for every edge
- numbers: optimal order to search from top left to bottom right corner

<table>
<thead>
<tr>
<th>budget</th>
<th>convolutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>8 191</td>
</tr>
<tr>
<td>256</td>
<td>270 335</td>
</tr>
<tr>
<td>512</td>
<td>794 623</td>
</tr>
<tr>
<td>1 280</td>
<td>2 367 487</td>
</tr>
<tr>
<td>12 800</td>
<td>25 960 447</td>
</tr>
</tbody>
</table>
SOTA
An interactive service?

Requirements
- response time within 100 ms
- ideally possible on large networks... …or at least city areas (> 50k nodes)
SOTA
An interactive service?

Requirements
- response time within 100 ms
- ideally possible on large networks... 
  ... or at least city areas (> 50k nodes)

Reality
- response time within minutes, on really small networks
- memory consumption too large (≥ budget · n)
SOTA
An interactive service?

Requirements
- response time within 100 ms
- ideally possible on large networks... 
  ...or at least city areas (> 50k nodes)

Reality
- response time within minutes, on really small networks
- memory consumption too large (≥ budget \cdot n)

Lost Cause?
- even with optimal processing: only small networks possible
- response times too large for interactive online service
SOTA
A Possible Solution

Exactness
- computation only exact within bounds of an inexact model
- large amount of convolutions for little improvement
  (looks back really far)
SOTA
A Possible Solution

Exactness
- computation only exact within bounds of an inexact model
- large amount of convolutions for little improvement
  (looks back really far)

Reality
- road networks contain only limited sets of viable routes
- no one will ever turn back for a few hundred miles after experiencing a stop and go
SOTA
A Possible Solution

Exactness
- computation only exact within bounds of an inexact model
- large amount of convolutions for little improvement
  (looks back really far)

Reality
- road networks contain only limited sets of viable routes
- no one will ever turn back for a few hundred miles after experiencing a stop and go

Our Goals (ongoing research)
- extract sparse but meaningful subgraphs from road network
- only evaluate stochastic model to provide high quality routes
Outline

Introduction

Motivation

Preliminaries - Contraction Hierarchies

Alternative Routes

Route Corridors

Final Remarks
Contraction Hierarchies

Method
- n-level preprocessing method for static routeplanning
- bidirectional search, only upwards
- introduces shortcuts to represent unique shortest paths

Properties
- preprocessing within minutes
- query times below a millisecond
- next to no memory overhead at all
- very small search space (around a few hundred nodes)
Contraction Hierarchies

Method
- n-level preprocessing method for static routeplanning
- bidirectional search, only upwards
- introduces shortcuts to represent unique shortest paths

Properties
- preprocessing within minutes
- query times below a millisecond
- next to no memory overhead at all
- very small search space (around a few hundred nodes)
Contraction Hierarchies

Method
- n-level preprocessing method for static route planning
- bidirectional search, only upwards
- introduces shortcuts to represent unique shortest paths

Properties
- preprocessing within minutes
- query times below a millisecond
- next to no memory overhead at all
- very small search space (around a few hundred nodes)
Method
- n-level preprocessing method for static routeplanning
- bidirectional search, only upwards
- introduces shortcuts to represent unique shortest paths

Properties
- preprocessing within minutes
- query times below a millisecond
- next to no memory overhead at all
- very small search space (around a few hundred nodes)
Contraction Hierarchies

Search Space

Illustration based on OpenStreetMap graph of Baden-Wuerttemberg
www.openstreetmap.org
Contraction Hierarchies

Search Space

contracted search space

extracted search space

*Illustration based on OpenStreetMap graph of Baden-Wuerttemberg*

www.openstreetmap.org
Contraction Hierarchies

Path Extraction

Methods

- recursive
  - store middle node for each shortcut
    → find composing edges
  - store composing edge ids
    → much faster, double the overhead

- store fully expanded shortest path
  - costly overhead, but fastest method
  - can be optimized by storing pointers into longer paths
Definition

For a graph $G = (V, A)$, a shortest path $P = \langle s, \ldots, t \rangle$ a path $\tilde{P}$ is a viable alternative path if it fulfils the following conditions:

1. bounded stretch
   \[ |\tilde{P}| \leq \alpha |P| \]
2. limited sharing
   \[ |P \cap \tilde{P}| \leq \beta |P| \]
3. local optimality
   \[ \text{any } p \subseteq \tilde{P} \text{ with } |p| \leq \gamma |P| \text{ is optimal} \]

Where $\alpha (>1)$, $\beta (<1)$, $\gamma (<1)$ can be chosen as desired.
Alternative Route Calculation

Alternative Graphs
- iterative approach
- penalization of calculated path / surrounding edges
  → needs fully dynamic searches, e.g. Dijkstra’s algorithm
  → currently too slow for interactive process
- creates full graph from which alternative routes can be extracted

Via Node / Plateau
- composed path \( \tilde{P} = \langle s, \ldots, v \rangle \langle v, \ldots, t \rangle \)
- possible sources of \( v \):
  1. additional meeting nodes within the CH search space
  2. set of precomputed candidates for groups of nodes
- test different \( v \) until applicable alternative route found
Alternative Routes

Testing Criteria

Bounded Stretch

- heuristic: $d(s, v) \in SP(s) + d(v, t) \in SP(t)$
  (may use incorrect distance labels)
- full check: compute $d(s, v)$ and $d(v, t)$

Limited Sharing

- heuristic: comparison based on shortcuts
  (shortcuts might share parts of paths)
- full check: check on extracted paths

Local Optimality

- heuristic: check whether $p(v - \gamma |P|, v + \gamma |P|)$ is a shortest path
  (2 approximation)
- full check: quadratic number of shortest path queries
  (too expansive)
Alternative Route Calculation

Clusters

Precomputing Via Nodes

- only few reasonable routes between regions
  - small sets of possible via node candidates
  - allows for per region pair precomputation

- expansive admissibility tests only on candidate set for regions
  - fast: only few, good candidates to be tested
  - increased success rate: candidate need not be in search space intersection

Numbers (Europe, XX regions)

- avg. via nodes between region pairs: 12.2
- precomputation time: 4.3 hours
- query time: 0.1 ms
Alternative Routes
Multiple Alternatives

Procedure

- iterate on further via nodes
- candidate sets for each further alternative
- criteria to be tested against all previously found alternatives

Results

<table>
<thead>
<tr>
<th>n</th>
<th>success rate [%]</th>
<th>calculation time [ms]</th>
<th>candidates</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>81.2</td>
<td>0.1</td>
<td>12.2</td>
</tr>
<tr>
<td>2</td>
<td>51.2</td>
<td>0.3</td>
<td>15.0</td>
</tr>
<tr>
<td>3</td>
<td>25.0</td>
<td>0.4</td>
<td>14.2</td>
</tr>
</tbody>
</table>
Route Corridors

Concept
- developed to make hybrid route planning robust on a mobile device
- idea: not only transmit shortest path, transmit additional information close to it
Route Corridors

Concept
- developed to make hybrid route planning robust on a mobile device
- idea: not only transmit shortest path, transmit additional information close to it

Realizations

Turn Corridor
- initialize with shortest path
- allow for one deviation
- fill in shortest path information
- repeat

Deviation-Time Corridor
- initialize with shortest path
- include everything reachable within a deviation budget
- fill in shortest path information
Corridor graphs

Computation

**Algorithm** (iterative)

- grow a shortest path tree to \( t \) (linear scans from top to bottom)
- extract what is needed for the corridor (two stack approach)
- only augment if information is missing (prune search space)

**Exploitable CH Properties**

- \( v \in SP(s) \rightarrow SP(v) \subseteq SP(s) \)
- \( SP(s) \) and \( SP(t) \) contain all information for any shortest path \( P(v, t), v \in SP(s) \)
- for any shortest path \( P = \langle s = v_0, \ldots, v_k = t \rangle \) holds:
  \[ \exists v_i : \langle v_0, \ldots, v_i \rangle \subset SP(s) \land \langle v_i, \ldots, v_k \rangle \subset SP(t) \]
Corridor Graphs
Shortest Path Trees

Initialization Phase
- single (complete) backwards search
- will result in some correct
  ... and some incorrect distance labels

Sweep Phase
- extract forward search space
  (one or many nodes)
- sweep search space from top to bottom
- distance values now correct for any \( v \) within the swept search space
Corridor Graphs
Shortest Path Trees

Initialization Phase
- single (complete) backwards search
- will result in some correct
  ... and some incorrect distance labels

Sweep Phase
- extract forward search space
  (one or many nodes)
- sweep search space from top to bottom
- distance values now correct for any \( v \)
  within the swept search space
**Two Stack Method**

- initialize $S_0 = s$, $S_1 = \emptyset$
- follow parent pointers of topmost element from $S_0$
  - deviation vertices pushed on $S_1$
  - parent vertex pushed on $S_0$
- when $S_0$ runs empty swap stacks
  - extend tree if necessary
  - repeat extraction
- works as close to target as possible
  → cache efficient
Corridor Graphs

Early pruning

Extraction Based Pruning

- path extraction increases number of nodes shortest path distances are known for
  - search space generation able to stop on known distance
  - only small search spaces generated

- computation of deviation vertices depends on extracted paths
  - path extraction necessary anyhow
  - allows for early pruning, for free
Corridor Graphs

Results

Computation

- input graph: European road network
- taken from 9th DIMACS implementation challenge (2006)
- edge based version with (high) turn penalties

<table>
<thead>
<tr>
<th>turns</th>
<th>time [ms]</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.73</td>
<td>1351</td>
</tr>
<tr>
<td>1</td>
<td>5.67</td>
<td>4835</td>
</tr>
<tr>
<td>2</td>
<td>9.73</td>
<td>12204</td>
</tr>
<tr>
<td>3</td>
<td>16.26</td>
<td>25892</td>
</tr>
</tbody>
</table>

Quality

- random *drives* with fixed failure rate
- next to *perfect success rates* for three turns and reasonable failure rates
Conclusion

On The Road to Interactive SOTA (₹)
- full search to slow and memory inefficient
- evaluation on sparse subgraphs might form a valid alternative
- existing techniques could be a good starting point
  - alternative routes
  - alternative graphs – if possible to compute fast
  - corridor graphs
  - arbitrary combinations

Ongoing Research
- fast calculation of alternative graphs
- a different approach to alternative routes
- SOTA on sparse subgraphs of large networks