

Review of “Finding Fastest Paths on A Road Network with Speed Patterns”

Matthias Frei

ETH Zürich

May 8, 2010

Outline

- 1 Introduction
- 2 Problem Definition
 - Network Model
 - Fastest Path Queries
- 3 Fastest Path Computation
 - Overview
 - Travel Time
 - The IntAllFastestPaths Algorithm
- 4 Example
- 5 Conclusion

Outline

- 1 Introduction
- 2 Problem Definition
 - Network Model
 - Fastest Path Queries
- 3 Fastest Path Computation
 - Overview
 - Travel Time
 - The IntAllFastestPaths Algorithm
- 4 Example
- 5 Conclusion

Introduction

- Car navigation systems usually use *shortest* path when calculating a route.

Introduction

- Car navigation systems usually use *shortest* path when calculating a route.
- A bit better: shortest path on a graph with edge length depending on average velocity on edge

Introduction

- Car navigation systems usually use *shortest* path when calculating a route.
- A bit better: shortest path on a graph with edge length depending on average velocity on edge
- Shortest path $\neq$ *fastest* path

Introduction

- Car navigation systems usually use *shortest* path when calculating a route.
- A bit better: shortest path on a graph with edge length depending on average velocity on edge
- Shortest path $\neq$ *fastest* path
- Several situations in which shortest path not optimal route
 - rush-hour traffic
 - holiday traffic
 - accidents, road works
 - ...
- Using average velocities won't fix this

Introduction (2)

- Ideas to do better:

Introduction (2)

- Ideas to do better:
 - Use time dependent velocities for the edges.

Introduction (2)

- Ideas to do better:
 - Use time dependent velocities for the edges.
 - Use statistical data for edge velocities $\Rightarrow$ velocity-“patterns”.
(In particular, we will use *piecewise constant* functions for the velocity on the edges)

Introduction (2)

- Ideas to do better:
 - Use time dependent velocities for the edges.
 - Use statistical data for edge velocities $\Rightarrow$ velocity-“patterns”.
(In particular, we will use *piecewise constant* functions for the velocity on the edges)
 - Categorize days such that for to days in the same category, we can use the same velocity-pattern

Introduction (2)

- Ideas to do better:
 - Use time dependent velocities for the edges.
 - Use statistical data for edge velocities $\Rightarrow$ velocity-“patterns”.
(In particular, we will use *piecewise constant* functions for the velocity on the edges)
 - Categorize days such that for too days in the same category, we can use the same velocity-pattern
- Of course this won't fix problems e.g. due to accidents
- At least we can use knowledge about regular changes of the velocity

Outline

- 1 Introduction
- 2 Problem Definition
 - Network Model
 - Fastest Path Queries
- 3 Fastest Path Computation
 - Overview
 - Travel Time
 - The IntAllFastestPaths Algorithm
- 4 Example
- 5 Conclusion

Outline

- 1 Introduction
- 2 Problem Definition
 - Network Model
 - Fastest Path Queries
- 3 Fastest Path Computation
 - Overview
 - Travel Time
 - The IntAllFastestPaths Algorithm
- 4 Example
- 5 Conclusion

Definition: Category Set

A (day-) category set D is a set of categories such that each day belongs to exactly one category in D .

Definition: CapeCod Pattern

Given a category set D , a **categorized piecewise constant speed** (CapeCod) pattern consists of one piecewise constant speed function s_d

$$s_d : T \rightarrow \mathbb{R}^+$$

for each category $d \in D$, where T is the time of day.

CapeCod?

- Categorized piecewise constant speed $\Rightarrow$ “CapeCod”?

CapeCod?

- Categorized piecewise constant speed $\Rightarrow$ “CapeCod”?
- The authors of the paper come from Boston, Massachusetts
- “Cape Cod” is a peninsula in eastern Massachusetts

CapeCod?

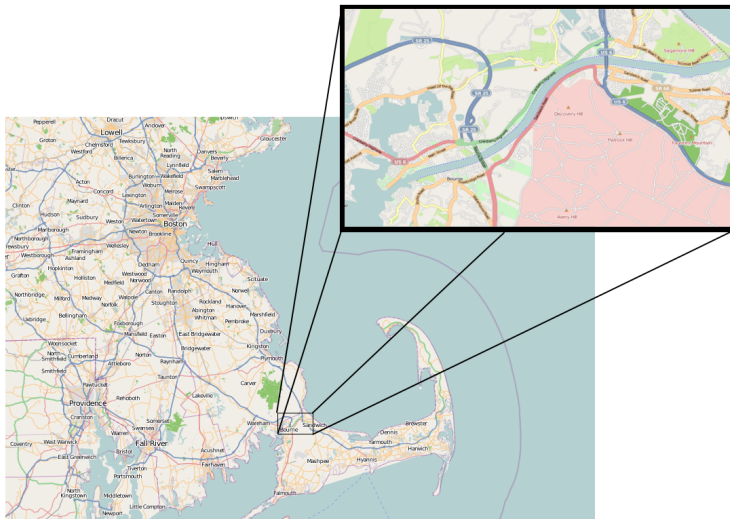
- Categorized piecewise constant speed $\Rightarrow$ “CapeCod”?
- The authors of the paper come from Boston, Massachusetts
- “Cape Cod” is a peninsula in eastern Massachusetts

“The Cape’s small-town character and large beachfront attract heavy tourism during the summer months.”

“The limited number of access points to the peninsula can result in large traffic backups during the tourist season.”¹

¹Source: http://en.wikipedia.org/wiki/Cape_cod

Cape Cod!



Source: <http://www.openstreetmap.org>

Matthias Frei (ETH Zürich)

Review of "Finding Fastest Paths on A Road

May 8, 2010

10 / 43

CapeCod Network (2)

Definition: CapeCod Network

Given a category set D , a CapeCod network is a directed graph $G(N, E)$. $N = \{(n_i, loc_i) | i \in 1, \dots, m\}$ is the set of nodes with loc_i their spatial locations. $E = \{(n_i, n_j, d_{i,j}, p_{i,j}) | i, j \in 1, \dots, m\}$ is the set of edges $n_i \rightarrow n_j$ where $d_{i,j}$ is the distance and $p_{i,j}$ is the CapeCod pattern assigned to the edge.

CapeCod Network (3) - Example

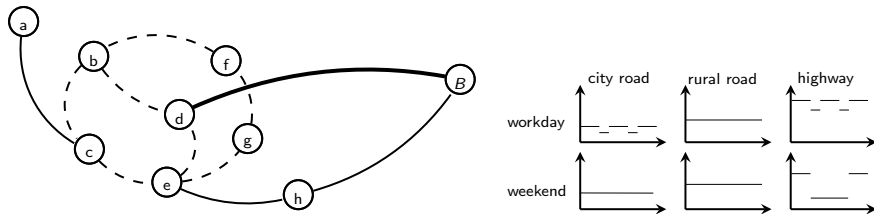


Figure: An example CapeCod network. The dashed lines represent city roads, the thin lines represent rural roads and the bold one represents a highway.

Outline

- 1 Introduction
- 2 Problem Definition
 - Network Model
 - Fastest Path Queries
- 3 Fastest Path Computation
 - Overview
 - Travel Time
 - The IntAllFastestPaths Algorithm
- 4 Example
- 5 Conclusion

Fastest Path Queries

- First idea: soonest possible arriving time

Fastest Path Queries

- First idea: soonest possible arriving time
- Two types of more general queries:
 - single Fastest Path query (“singleFP”)
 - all Fastest Path query (“allFP”)

Fastest Path Queries

- First idea: soonest possible arriving time
- Two types of more general queries:
 - single Fastest Path query (“singleFP”)
 - all Fastest Path query (“allFP”)
- ... but we will see that they can be solved using the same algorithm.

Fastest Path Queries (2) - The singleFP Query

Definition: singleFP Query

Given a CapeCod network $G(N, E)$, a start node $s \in N$, an end node $e \in N$, and a leaving time interval I , the *single fastest paths* query returns a time instant $t \in I$ and the corresponding fastest path from s to e such that leaving from s at time t minimizes travel time from s to e .

Fastest Path Queries (2) - The singleFP Query

Definition: singleFP Query

Given a CapeCod network $G(N, E)$, a start node $s \in N$, an end node $e \in N$, and a leaving time interval I , the *single fastest paths* query returns a time instant $t \in I$ and the corresponding fastest path from s to e such that leaving from s at time t minimizes travel time from s to e .

Example:

"I want to go from my house near node e to the beach at node B on Saturday, leaving sometime between 6am and 10am. Give me the route and the leaving time such that the journey is as short as possible."

Fastest Path Queries (3) - The allFP Query

Definition: allFP Query

Given a CapeCod network $G(N, E)$, a start node $s \in N$, an end node $e \in N$, and a leaving time interval I , the *all fastest paths* query returns a full partitioning of $I : I_1, \dots, I_k$ where each sub-interval is associated with a fastest path such that two leaving time instants in the same sub-interval lead to the same fastest path and two leaving time instants in two adjacent sub-intervals lead to different fastest paths.

Fastest Path Queries (3) - The allFP Query

Definition: allFP Query

Given a CapeCod network $G(N, E)$, a start node $s \in N$, an end node $e \in N$, and a leaving time interval I , the *all fastest paths* query returns a full partitioning of $I : I_1, \dots, I_k$ where each sub-interval is associated with a fastest path such that two leaving time instants in the same sub-interval lead to the same fastest path and two leaving time instants in two adjacent sub-intervals lead to different fastest paths.

Example:

"I want to go from my house near node e to the beach at node B on Saturday. Give me a listing which tells me the fastest route for leaving time between 6am and 10am."

Outline

- 1 Introduction
- 2 Problem Definition
 - Network Model
 - Fastest Path Queries
- 3 Fastest Path Computation
 - Overview
 - Travel Time
 - The IntAllFastestPaths Algorithm
- 4 Example
- 5 Conclusion

Outline

- 1 Introduction
- 2 Problem Definition
 - Network Model
 - Fastest Path Queries
- 3 Fastest Path Computation**
 - Overview**
 - Travel Time
 - The IntAllFastestPaths Algorithm
- 4 Example
- 5 Conclusion

Fastest Path Computation

- Conceptually similar to Shortest Path Computation

Fastest Path Computation

- Conceptually similar to Shortest Path Computation
- Difference: want to minimize *travel time* in contrast to *distance*
- Single time instant queries (i.e. leaving time is a scalar value) can easily be solved using normal shortest path algorithms

Fastest Path Computation

- Conceptually similar to Shortest Path Computation
- Difference: want to minimize *travel time* in contrast to *distance*
- Single time instant queries (i.e. leaving time is a scalar value) can easily be solved using normal shortest path algorithms
- Problems for singleFP and allFP queries:

Fastest Path Computation

- Conceptually similar to Shortest Path Computation
- Difference: want to minimize *travel time* in contrast to *distance*
- Single time instant queries (i.e. leaving time is a scalar value) can easily be solved using normal shortest path algorithms
- Problems for singleFP and allFP queries:
 - Leaving time is *not* a single value but an interval

Fastest Path Computation

- Conceptually similar to Shortest Path Computation
- Difference: want to minimize *travel time* in contrast to *distance*
- Single time instant queries (i.e. leaving time is a scalar value) can easily be solved using normal shortest path algorithms
- Problems for singleFP and allFP queries:
 - Leaving time is *not* a single value but an interval
 - For different leaving times, there are different optimal paths

Fastest Path Computation (2) - Approaches

Possible approaches:

Fastest Path Computation (2) - Approaches

Possible approaches:

- Discrete time model
 - Compute fastest path for many different leaving times

Fastest Path Computation (2) - Approaches

Possible approaches:

- Discrete time model
 - Compute fastest path for many different leaving times
 - $\Rightarrow$ Not exact enough

Fastest Path Computation (2) - Approaches

Possible approaches:

- Discrete time model
 - Compute fastest path for many different leaving times
 - $\Rightarrow$ Not exact enough
- Continuous time model
 - Consider travel time to be a *function* of the leaving time

Fastest Path Computation (2) - Approaches

Possible approaches:

- Discrete time model
 - Compute fastest path for many different leaving times
 - $\Rightarrow$ Not exact enough
- Continuous time model
 - Consider travel time to be a *function* of the leaving time
 - Exact solution

Fastest Path Computation (2) - Approaches

Possible approaches:

- Discrete time model
 - Compute fastest path for many different leaving times
 - $\Rightarrow$ Not exact enough
- Continuous time model
 - Consider travel time to be a *function* of the leaving time
 - Exact solution
 - $\Rightarrow$ Approach chosen here

Fastest Path Computation (2) - Approaches

Possible approaches:

- Discrete time model
 - Compute fastest path for many different leaving times
 - $\Rightarrow$ Not exact enough
- Continuous time model
 - Consider travel time to be a *function* of the leaving time
 - Exact solution
 - $\Rightarrow$ Approach chosen here

Outline

- 1 Introduction
- 2 Problem Definition
 - Network Model
 - Fastest Path Queries
- 3 Fastest Path Computation**
 - Overview
 - Travel Time**
 - The IntAllFastestPaths Algorithm
- 4 Example
- 5 Conclusion

Travel Time

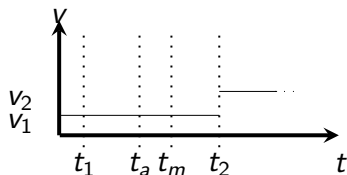
- Travel time is the time needed to travel along a given path in a CapeCod network, driving at exactly the speed given by the CapeCod patterns of the particular edges and the current time.

Travel Time

- Travel time is the time needed to travel along a given path in a CapeCod network, driving at exactly the speed given by the CapeCod patterns of the particular edges and the current time.
- The travel time is a function of the *leaving time*.

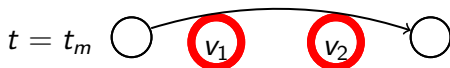
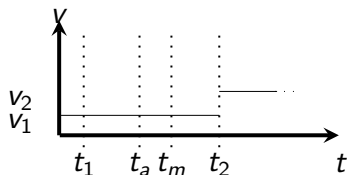
Travel Time

- Travel time is the time needed to travel along a given path in a CapeCod network, driving at exactly the speed given by the CapeCod patterns of the particular edges and the current time.
- The travel time is a function of the *leaving time*.
- Example: Derive the travel time along a single edge:



Travel Time

- Travel time is the time needed to travel along a given path in a CapeCod network, driving at exactly the speed given by the CapeCod patterns of the particular edges and the current time.
- The travel time is a function of the *leaving time*.
- Example: Derive the travel time along a single edge:



Travel Time (2)

- t_a is the time until which the whole edge can be traversed at speed v_1 .

$$t_a = t_2 - \frac{d}{v_1}$$

Travel Time (2)

- t_a is the time until which the whole edge can be traversed at speed v_1 .

$$t_a = t_2 - \frac{d}{v_1}$$

- The travel time along the edge, starting at time t is given by

$$T(t) = \begin{cases} \frac{d}{v_1} & \text{if } t \leq t_2 - \frac{d}{v_1} \\ (t_2 - t) + \frac{d - v_1(t_2 - t)}{v_2} & \text{if } t \in [t_2 - \frac{d}{v_1}, t_2] \\ \frac{d}{v_2} & \text{if } t \geq t_2 \end{cases}$$

Travel Time (2)

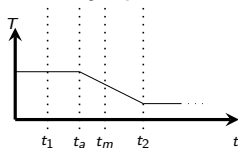
- t_a is the time until which the whole edge can be traversed at speed v_1 .

$$t_a = t_2 - \frac{d}{v_1}$$

- The travel time along the edge, starting at time t is given by

$$T(t) = \begin{cases} \frac{d}{v_1} & \text{if } t \leq t_2 - \frac{d}{v_1} \\ (t_2 - t) + \frac{d - v_1(t_2 - t)}{v_2} & \text{if } t \in [t_2 - \frac{d}{v_1}, t_2] \\ \frac{d}{v_2} & \text{if } t \geq t_2 \end{cases}$$

- The resulting function is always *piecewise linear*!



Travel Time (3)

We need travel time along a *path* not along an edge!

Travel Time (3)

We need travel time along a *path* not along an edge!

Assume that the following is given:

- Travel time $T(s \Rightarrow n_i, t \in I)$ of a path from s to n_i
- Travel time $T(n_i \rightarrow n_j, t)$ of the edge from n_i to n_j

How can we construct the travel time $T(s \Rightarrow n_j, t \in I)$?

Travel Time (3)

We need travel time along a *path* not along an edge!

Assume that the following is given:

- Travel time $T(s \Rightarrow n_i, t \in I)$ of a path from s to n_i
- Travel time $T(n_i \rightarrow n_j, t)$ of the edge from n_i to n_j

How can we construct the travel time $T(s \Rightarrow n_j, t \in I)$?

$$T(s \Rightarrow n_j, t \in I) = T(s \Rightarrow n_i, t) + T(n_i \rightarrow n_j, t + T(s \Rightarrow n_i, t))$$

Travel Time (3)

We need travel time along a *path* not along an edge!

Assume that the following is given:

- Travel time $T(s \Rightarrow n_i, t \in I)$ of a path from s to n_i
- Travel time $T(n_i \rightarrow n_j, t)$ of the edge from n_i to n_j

How can we construct the travel time $T(s \Rightarrow n_j, t \in I)$?

$$T(s \Rightarrow n_j, t \in I) = T(s \Rightarrow n_i, t) + T(n_i \rightarrow n_j, t + T(s \Rightarrow n_i, t))$$

It can be shown that the resulting function is still piecewise linear.

Outline

- 1 Introduction
- 2 Problem Definition
 - Network Model
 - Fastest Path Queries
- 3 Fastest Path Computation**
 - Overview
 - Travel Time
 - The IntAllFastestPaths Algorithm**
- 4 Example
- 5 Conclusion

The IntAllFastestPaths Algorithm - Overview

Consider a query given by a CapeCod network $G(N, E)$ a start node s , an end node e and a leaving time interval I .

The IntAllFastestPaths Algorithm - Overview

Consider a query given by a CapeCod network $G(N, E)$ a start node s , an end node e and a leaving time interval I .

The main ideas of the IntAllFastestPaths Algorithm are:

The IntAllFastestPaths Algorithm - Overview

Consider a query given by a CapeCod network $G(N, E)$ a start node s , an end node e and a leaving time interval I .

The main ideas of the IntAllFastestPaths Algorithm are:

(1) Priority Queue

Maintain a priority queue of paths $s \rightarrow \dots \rightarrow n_i$.

- Order paths by *minimum* of the function $T(n \Rightarrow n_i, t \in I) + T_{est}(n_i \Rightarrow e)$
- $T_{est}(n_i \Rightarrow e)$ is a lower bound estimator for the remaining travel time

Notation: $s \Rightarrow n_i$ is a shorthand for a specific path $s \rightarrow \dots \rightarrow n_i$.

(2) Path Expansion

In each step, we pick a path to expand from the priority queue.

- Pick the path $s \Rightarrow n_i$ whose functions minimum is minimal amongst all paths in the queue
- Calculate the travel time function $T(s \Rightarrow n_i \rightarrow n_j)$ for all neighbours n_j of n_i and put the paths $s \Rightarrow n_i \rightarrow n_j$ into the priority queue.

(3) Lower Border Function

Maintain a special travel time function $T_{lb}(t \in I)$ called the lower border function.

- The lower border of the travel time functions of all identified paths from s to e .
- Consists of multiple travel time functions, each corresponding to some path and a subinterval of I during which this path is the smallest.
- Represents the result of the allFP query

(4) Result

For the allFP query:

- Stop if either the priority queue is empty, ...
- ...or if the minimum of the function picked from the top of the queue is larger than the maximum of the lower border function.

(4) Result

For the allFP query:

- Stop if either the priority queue is empty, ...
- ...or if the minimum of the function picked from the top of the queue is larger than the maximum of the lower border function.

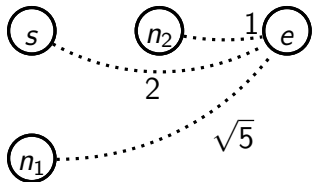
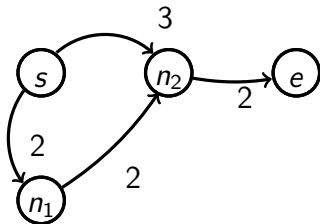
For the singleFP query:

- Stop as soon as some path ending in e is picked from the queue.

Outline

- 1 Introduction
- 2 Problem Definition
 - Network Model
 - Fastest Path Queries
- 3 Fastest Path Computation
 - Overview
 - Travel Time
 - The IntAllFastestPaths Algorithm
- 4 Example
- 5 Conclusion

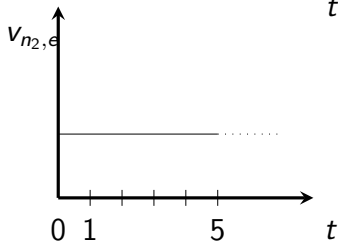
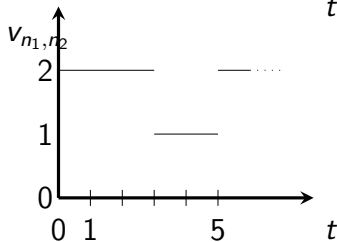
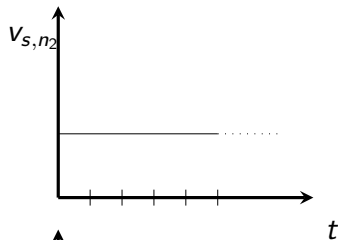
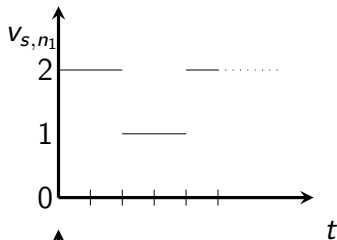
Example



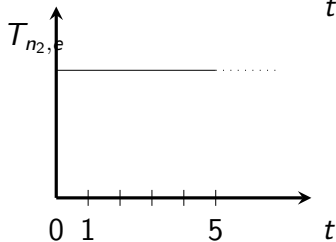
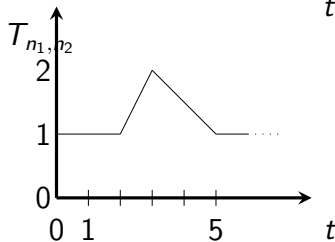
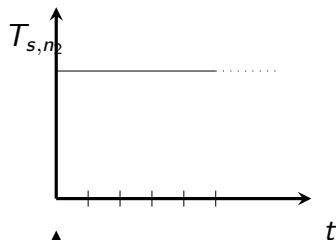
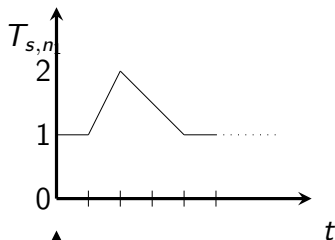
Query:

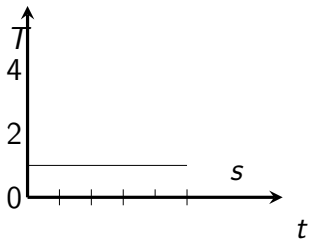
- singleFP and allFP query
- Start node: s
- End node: e
- Leaving time interval: $I = [0, 5]$

Example (2) - CapeCod Patterns

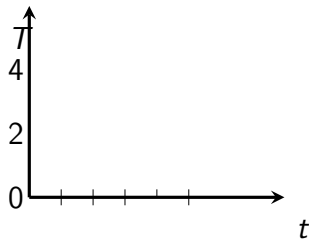


Example (3) - Edge Travel Times

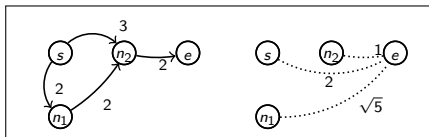


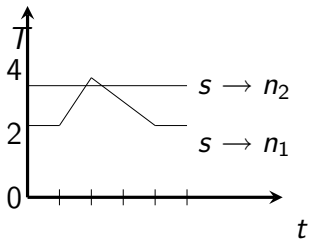


Priority Queue

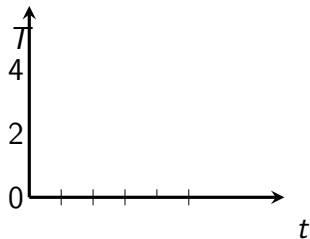


Lower Border

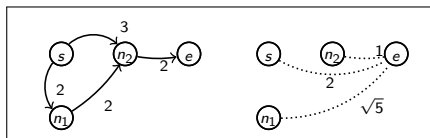


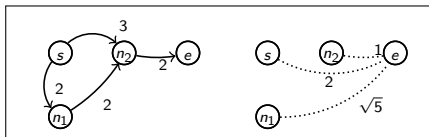
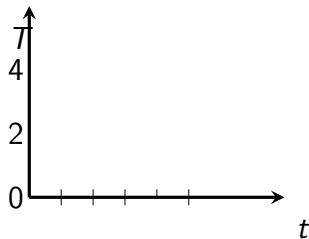
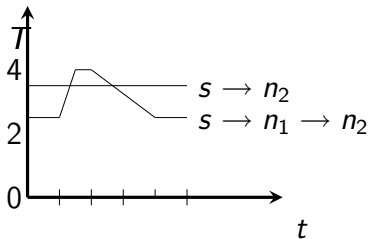


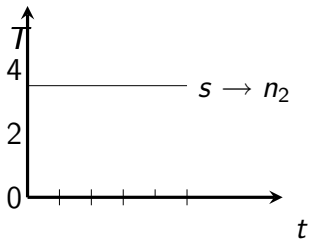
Priority Queue



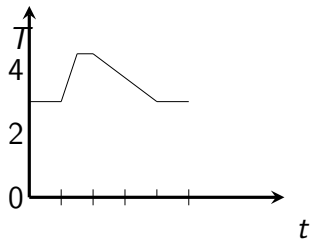
Lower Border



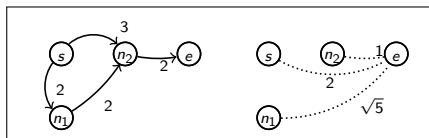


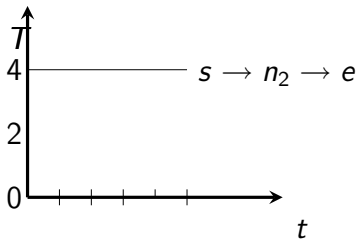


Priority Queue

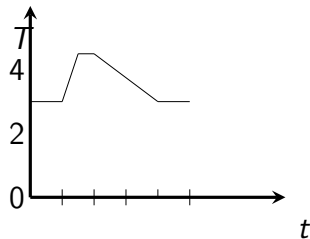


Lower Border

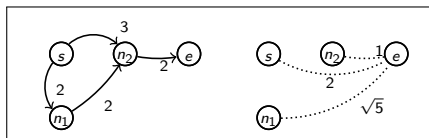


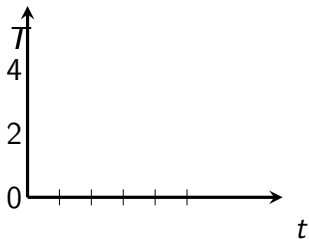


Priority Queue

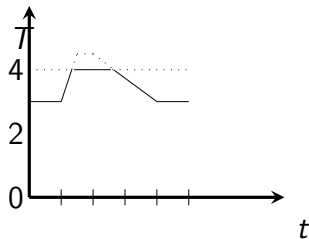


Lower Border

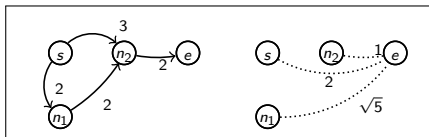




Priority Queue



Lower Border



Outline

- 1 Introduction
- 2 Problem Definition
 - Network Model
 - Fastest Path Queries
- 3 Fastest Path Computation
 - Overview
 - Travel Time
 - The IntAllFastestPaths Algorithm
- 4 Example
- 5 Conclusion

Conclusion

We saw

- ... the CapeCod network – a model for a graph with time dependant edge-velocities
- ... the singleFP and allFP queries – fastest path queries with non-scalar leaving values
- ... the IntAllFastestPaths algorithm – an algorithm to solve the proposed queries

Questions?