# Finding Fastest Paths on A Road Network with Speed Patterns

Evangelos Kanoulas, Yang Du, Tian Xia, and Donghui Zhang

College of Computer & Information Science
Northeastern University, Boston

September 29, 2006

Appeared in: International Conference on Data Engineering (ICDE06)

Presented by Kristian Torp

# Outline

# Problem

## Cause

Current navigation system assume that the travel time on a road segment is constant.

## Effect

The travel-time in for example rush hour is highly underestimated.

## Solution

"I may leave for work any time between 7am and 9am; please suggest all fastest paths, e.g. take route A if the leaving time is between 7 and 7:45, and take route B otherwise."

# Outline

# CapeCod Network

## Definition

Day Category $D$ is a day category set.

## Example

$D = \{workday, non - workday\}$

## Definition

CAtegorized PiecewisE COnstant speeD (CapeCod) CapeCod is a pattern of one daily speed pattern for every day-category in $D$.

## Example

$$\begin{aligned} \text{workday} = & [00:00 - 07:00) = 1.00 mp/h \\ & [07:00 - 09:00) = 0.75 mp/h \\ & [09:00 - 15:00) = 1.00 mp/h \\ & [15:00 - 17:00) = 0.60 mp/h \\ & [17:00 - 24:00) = 1.00 mp/h \end{aligned}$$

# CapeCod Network, cont.

## Definition

CapeCod Network A directed graph $G(N, E)$ where $N = \{(n_i, loc_i) \mid i \in [1, m]\}$ is the set of nodes and $E = \{n_i, n_j, d_{ij}, pat_{ij} \mid i, j \in [1, m]\}$ is the set of edges where $d_{ij}$ is a distance and $pat_{ij}$ is a CapeCod pattern.

# Queries

## Single Fastest Path (singleFP) Query

Given a start node $s$, an end node $e$, and a leaving time interval $I$, find the time instant $l \in I$ and the corresponding fastest path from $s$ to $e$ such that leaving from $s$ at time $l$ minimizes the travel time from $s$ to $e$

## All Fastest Path (allFP) Query

Given a CapeCod network, a start node $s$ and an end node $e$, and a leaving time interval $I$, find a full partitioning of $I$: $I_1$,... ,$I_k$, where each sub-interval is associated with a fastest path, such that two leaving time instants in one sub-interval leads to the same fastest-path and two leaving time instants in two adjacent sub-intervals leads to different fastest paths.

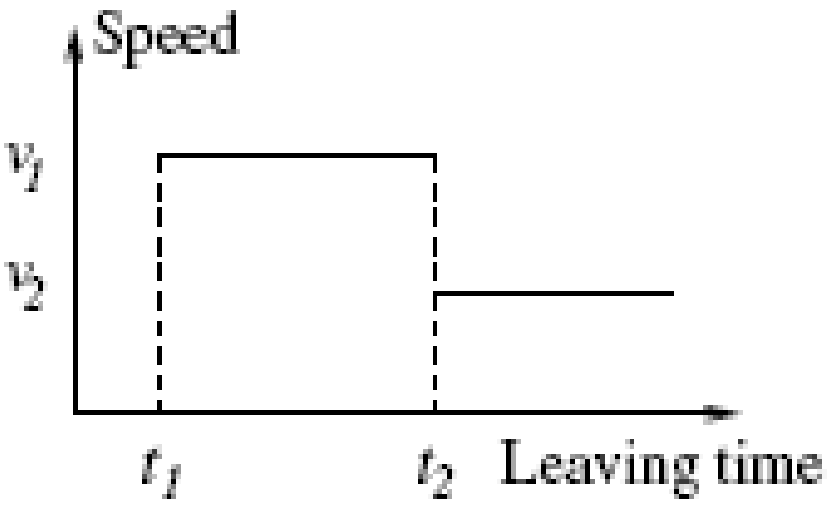
# Storage Model

- Connectivity Cluster Access Method (CCAM)
- For each node $n_i$
  - Stores the location $loc_i$
  - Stores the list of neighbors
  - Stores the Euclidian distance to each neighbor
  - Stores the CapeCod speed pattern ($pat_{ij}$) to each neighbor
- Hilbert curve value indexes the $loc_i$ value

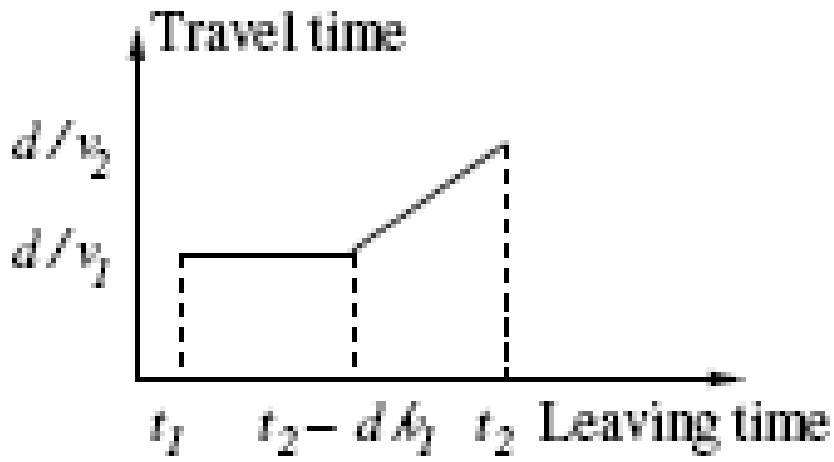
Operations

- $FindNode(n_i)$
- $GetSuccessor(n_i)$
- Modify the network

# From Speed Pattern to Travel Time




(a) Speed




(b) Travel time

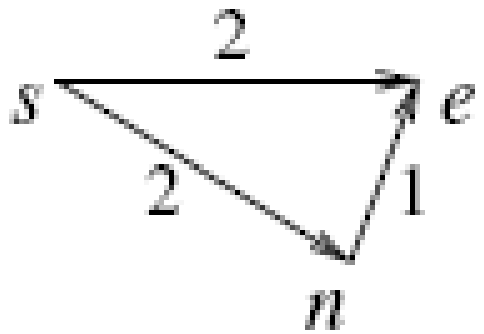
- Speed $v_1$ in $[t_1, t_2)$ and $v_2$ there after
- Speed can be changed more than once (not likely)

Travel-Time Function

$$T(l \in [t_1, t_2], n_i \rightarrow n_j) = \begin{cases} \frac{d}{v_1} & l \in [t_1, t_2, \frac{d}{v_1}] \\ (1 - \frac{v_1}{v_2})(t_2 - l) + \frac{d}{v_2} & l \in [t_2 - \frac{d}{v_1}, t_2] \end{cases}$$

# Outline

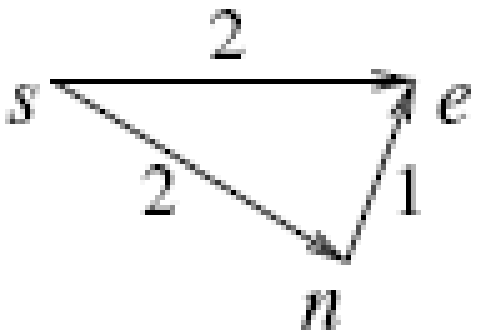
# Initialization




$s \longrightarrow e$: [6-8):1/3
$s \longrightarrow n$: [6-7):1/3, [7-8):1
$n \longrightarrow e$: [6-7:08):1/3, [7:08-8):1/10

$I = [6:50 - 7:05]$

# Initialization

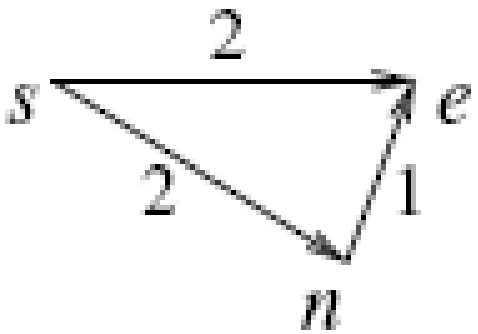



$s \longrightarrow e$: [6-8):1/3
$s \longrightarrow n$: [6-7):1/3, [7-8):1
$n \longrightarrow e$: [6-7:08):1/3, [7:08-8):1/10

$I = [6:50 - 7:05]$

$$T(I \in [6:50 - 7:05], s \longrightarrow e) =$$

# Initialization

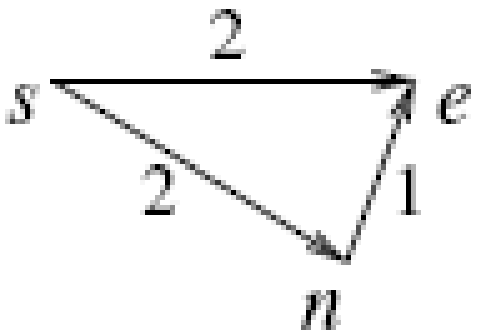



$s \longrightarrow e$: [6-8):1/3
$s \longrightarrow n$: [6-7):1/3, [7-8):1
$n \longrightarrow e$: [6-7:08):1/3, [7:08-8):1/10

$I = [6:50 - 7:05]$

$$T(I \in [6:50 - 7:05], s \longrightarrow e) = 6$$

# Initialization



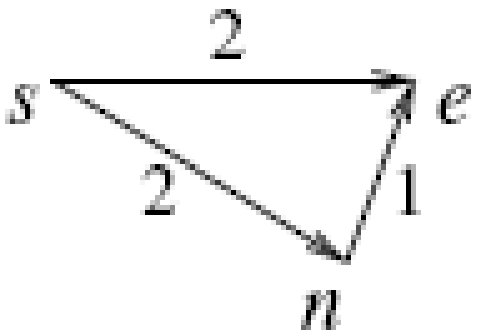

$s \longrightarrow e$: [6-8):1/3
$s \longrightarrow n$: [6-7):1/3, [7-8):1
$n \longrightarrow e$: [6-7:08):1/3, [7:08-8):1/10

$I = [6:50 - 7:05]$

$$T(I \in [6:50 - 7:05], s \longrightarrow e) = 6$$

$$T(I \in [6:50 - 7:05], s \longrightarrow n) = \left\{ \right.$$

# Initialization



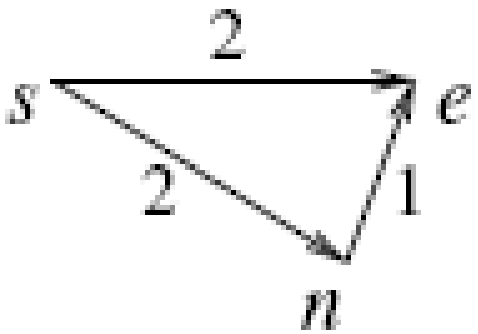

$s \longrightarrow e$: [6-8):1/3
$s \longrightarrow n$: [6-7):1/3, [7-8):1
$n \longrightarrow e$: [6-7:08):1/3, [7:08-8):1/10

$I = [6:50 - 7:05]$

$$T(I \in [6:50 - 7:05], s \longrightarrow e) = 6$$

$$T(I \in [6:50-7:05], s \longrightarrow n) = \begin{cases} 6 & I \in [6:50 - 6:54 \end{cases}$$

# Initialization



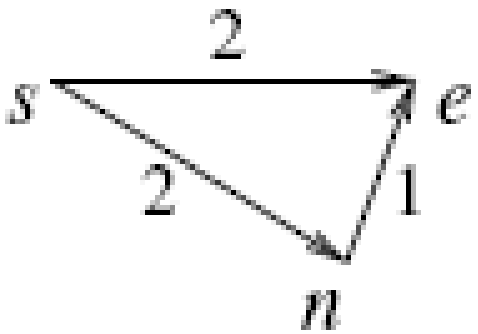

$s \longrightarrow e$: [6-8):1/3
$s \longrightarrow n$: [6-7):1/3, [7-8):1
$n \longrightarrow e$: [6-7:08):1/3, [7:08-8):1/10

$I = [6:50 - 7:05]$

$$T(I \in [6:50 - 7:05], s \longrightarrow e) = 6$$

$$T(I \in [6:50-7:05], s \longrightarrow n) = \begin{cases} 6 & I \in [6:50 - 6:54) \\ 2 & I \in [7:00 - 7:05) \end{cases}$$

# Initialization



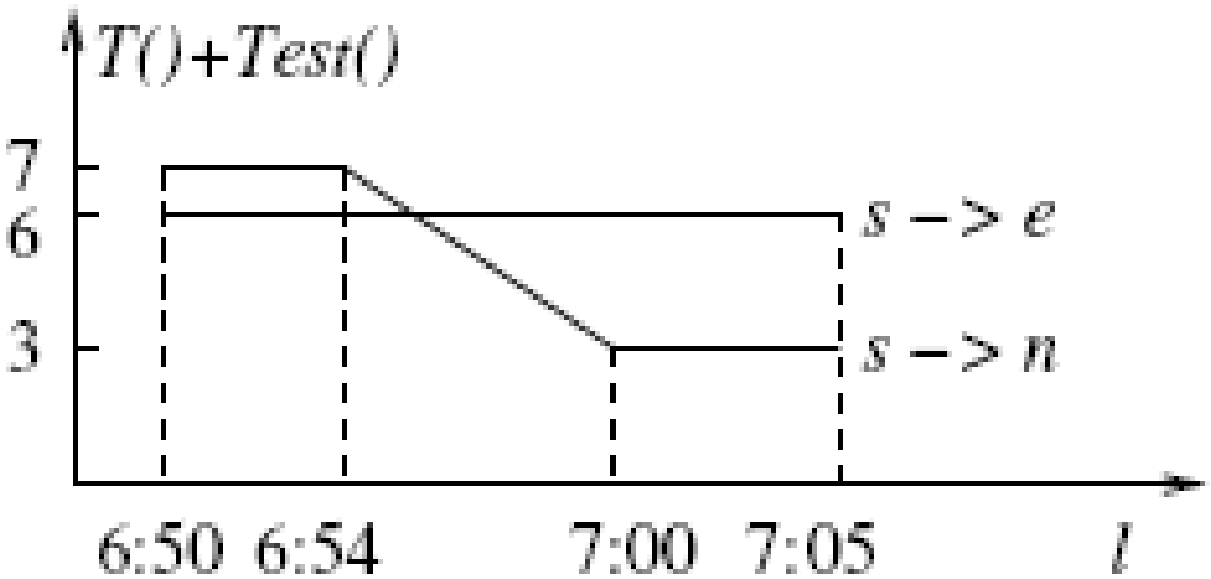

$s \longrightarrow e$: [6-8):1/3
$s \longrightarrow n$: [6-7):1/3, [7-8):1
$n \longrightarrow e$: [6-7:08):1/3, [7:08-8):1/10

$I = [6:50 - 7:05]$

$$T(I \in [6:50 - 7:05], s \longrightarrow e) = 6$$

$$T(I \in [6:50-7:05], s \longrightarrow n) = \begin{cases} 6 & I \in [6:50 - 6:54) \\ 2 & I \in [7:00 - 7:05) \\ \frac{2}{3}(7:00 - I) + 2 & I \in [6:54 - 7:00) \end{cases}$$

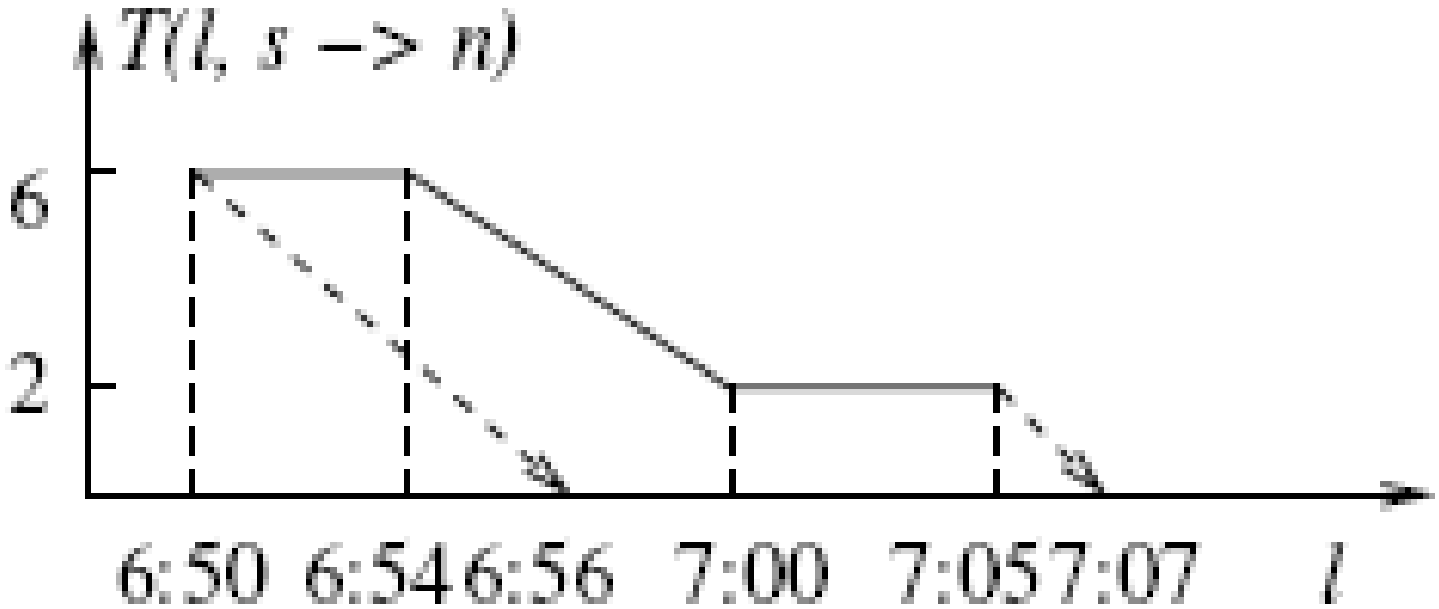
# Path Expansion




Based on $T() + T_{est}()$

- Must expand the path $s \longrightarrow n$

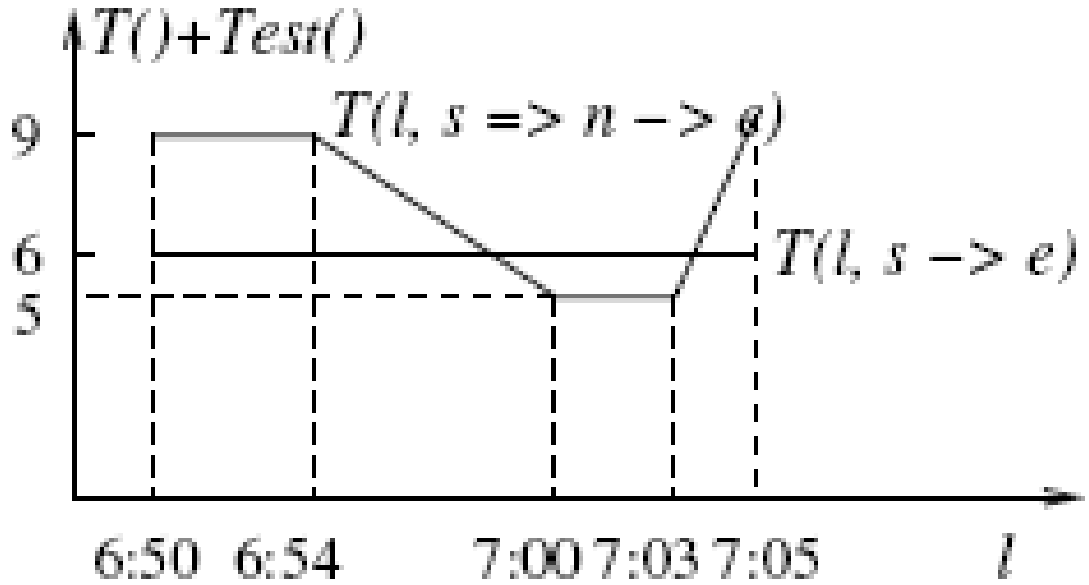
# Path Expansion, cont.




- Determine the interval during which the travel-time function for road segment is needed
- Determine the time instants at which the resulting function changes
- For each time interval determine the travel-time function

$$T(l \in [6:56{-}7:07], n \to e) = \begin{cases} 3 & l \in [6:56 - 7:05 \\ 10 - \frac{7}{3}(7:08 - l) & l \in [7:05 - 7:07 \end{cases}$$

# Fastest Path Result




$$T(l \in [6:50-7:05], s \to e) = \begin{cases} s \to e & l \in [6:50-6:58:30) \\ s \to n \to e & l \in [6:58:30-7:03: \\ s \to e & l \in [7:03:26-7:05) \end{cases}$$

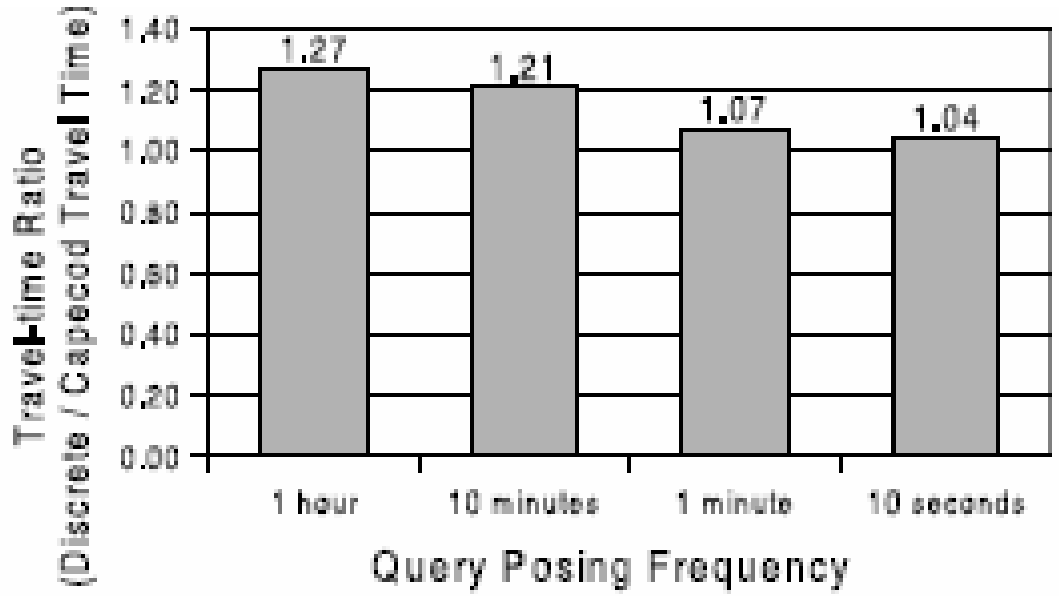
# Outline

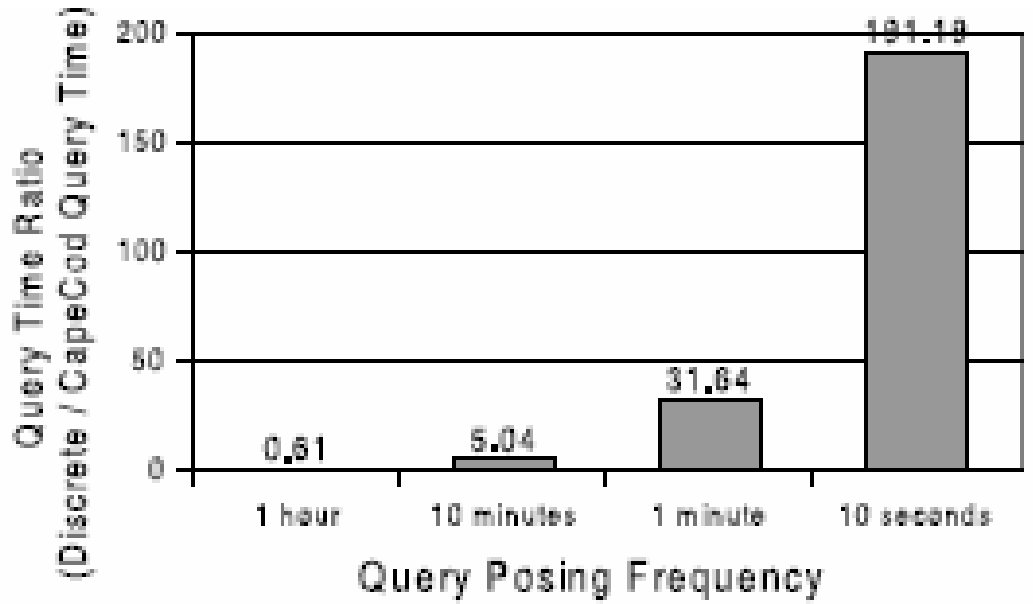
# Setup

- Road network : Suffolk county of Massachusetts (Boston)
  - 14,456 nodes and 20,461 directed edges
- Day categories $D$: $=\{$ workday and non-workday $\}$
- Road segments types
  - inbound highways
  - outbound highway
  - local roads outside Boston
  - local roads inside Boston
- Speed Pattern

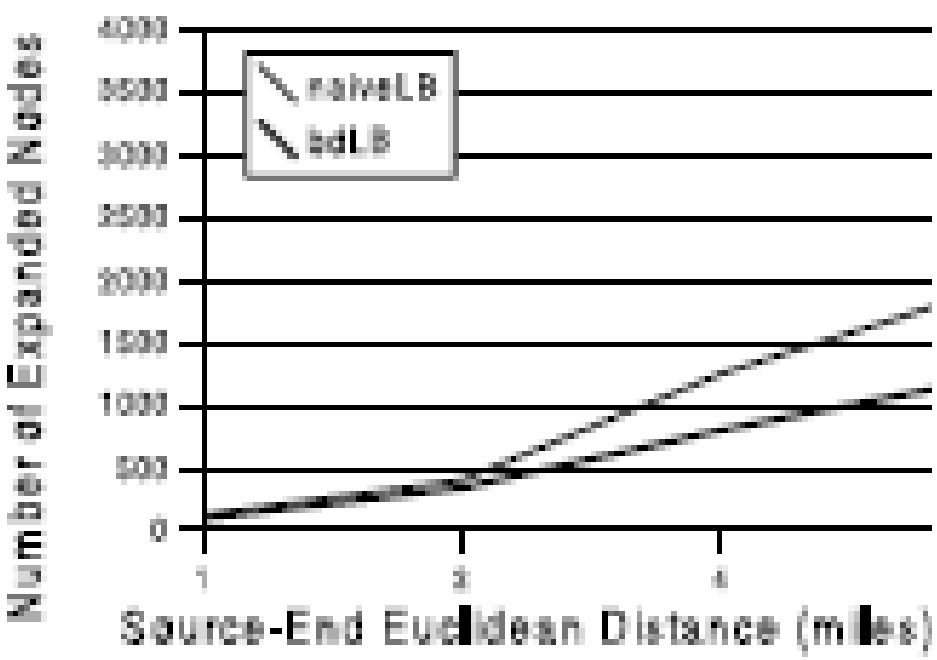
| | Inbound Hw | | Outbound Hw | | Local (in) | | Local (out) |
|---|---|---|---|---|---|---|---|
| non-workday | 65 | | 65 | | 40 | | 40 |
| workday | 00:00-07:00 | 65 | 00:00-04:00 | 65 | 00:00-04:00 | 40 | 40 |
| | 07:00-10:00 | 20 | 04:00-07:00 | 30 | 04:00-07:00 | 20 | |
| | 10:00-24:00 | 65 | 07:00-24:00 | 65 | 07:00-24:00 | 40 | |

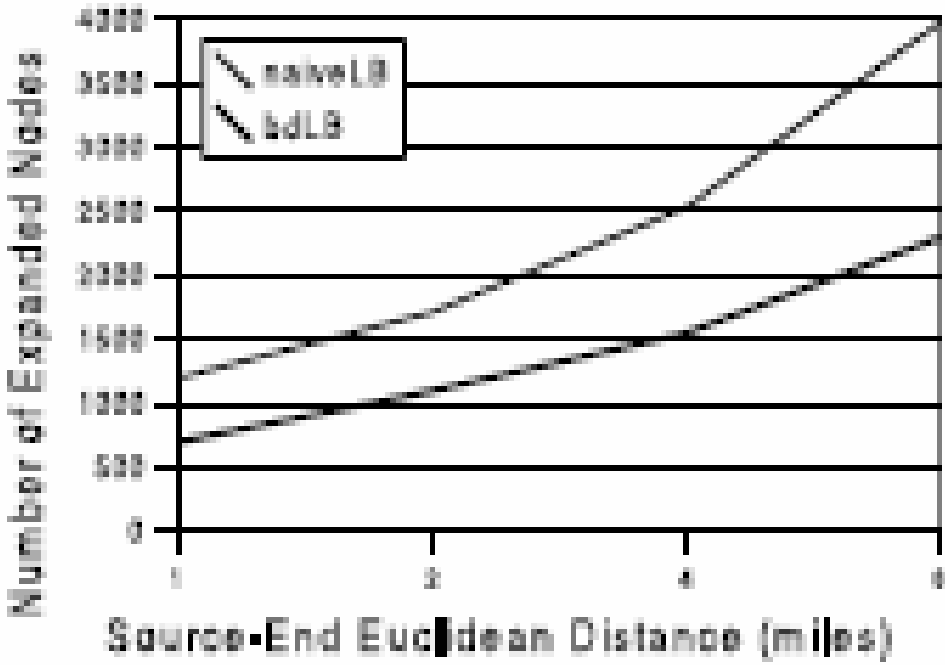
# Performance



# Performance, cont.



# Outline

# Strong Points

- Contributions of paper clearly pointed out
- Good coverage of related work
- Explanation of core algorithm in Section 4 including examples

# Weak Points

- Synthetic setup of CapeCod travel-time speed patterns
- Some references are pretty old ([5, 9, 10, 12])
- No validation of computed travel times to actual travel times