Getting from Here to There

Prof. David Bernstein

Department of Computer Science
James Madison University

17 February 2009
Longwood University

Some History

- First Generation:
  A map
- Second Generation:
  Add the current location
- Third Generation:
  Add directions

Required Technologies

- First Generation:

  Processor and storage
  Graphical display device (probably color)
  Input devices (e.g., keyboard, pointing device)
  Software for manipulating raster/vector data

Required Technologies

- First Generation:
  - Processor and storage
  - Graphical display device (probably color)
  - Input devices (e.g., keyboard, pointing device)
  - Software for manipulating raster/vector data
- Second Generation:
  - Positioning hardware (e.g., GPS, dead-reckoning)
  - Positioning software

Required Technologies

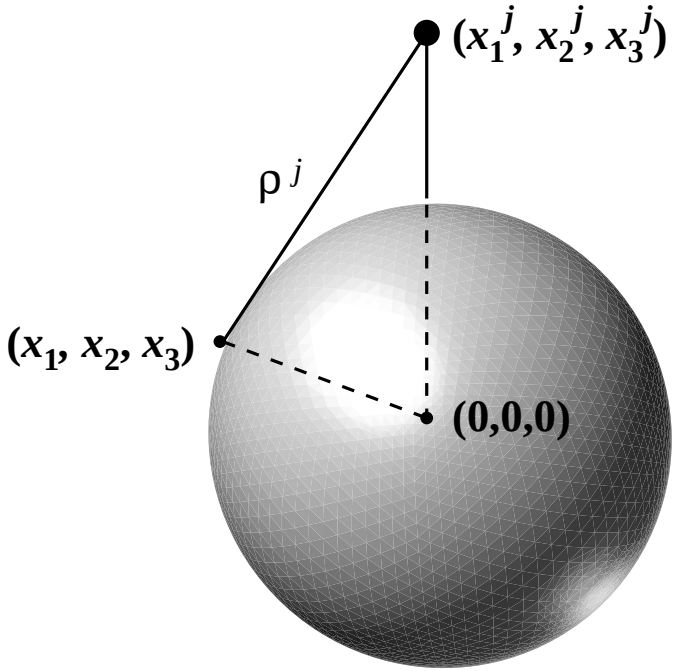
- First Generation:
  - Processor and storage
  - Graphical display device (probably color)
  - Input devices (e.g., keyboard, pointing device)
  - Software for manipulating raster/vector data
- Second Generation:
  - Positioning hardware (e.g., GPS, dead-reckoning)
  - Positioning software
- Third Generation:
  - Map-matching software
  - Route-finding software

Purposes of This Talk

- Describe some of the "classical" algorithms that have been used
- Describe some of my research in this area
- Make sure that you can, if you want, implement your own system
- Interest you in pursuing the "open questions" in this area

Global Positioning System Basics

- The Satellites:
  24 satellites in 6 evenly-spaced orbits with radii of 26,560km
  Each satellite circles the Earth every 11 hours and 58 minutes
- The Coordinate Systems:
  Geocentric (i.e., the origin is at the center of the Earth)



Triangulation with Perfect Clocks

- Notation:

  T^j is the "actual" time satellite j transmits

  T is the "actual" time that the signal is received

  $\Delta T^j \equiv T - T^j$

  c is the speed of light

- An Important Relationship:

  The distance is $c \cdot \Delta T^j$

Triangulation with Perfect Clocks (cont.)

The System to be Solved

$$\begin{aligned}
\sqrt{(x_1^i - x_1)^2 + (x_2^i - x_2)^2 + (x_3^i - x_3)^2} &= c \cdot \Delta T^i \\
\sqrt{(x_1^j - x_1)^2 + (x_2^j - x_2)^2 + (x_3^j - x_3)^2} &= c \cdot \Delta T^j \\
\sqrt{(x_1^k - x_1)^2 + (x_2^k - x_2)^2 + (x_3^k - x_3)^2} &= c \cdot \Delta T^k
\end{aligned}$$

Triangulation with Imperfect Clocks

- Two Observations:

  The time on the receiver's clock when the signal is received, t, differs from the "actual time" when the signal is received by the amount δ

  The clocks on the satellites are not perfect so $t^j = T^j + \delta^j$

- The Implication:

$$\Delta T^j \equiv T - T^j = (t - \delta) - (t^j - \delta^j) = (t - t^j) + (\delta^j - \delta) = \Delta t^j + (\delta^j - \delta)$$

Triangulation with Imperfect Clocks

The New System to be Solved

$$\begin{aligned}
\sqrt{(x_1^i - x_1)^2 + (x_2^i - x_2)^2 + (x_3^i - x_3)^2} &= c \cdot \Delta t^i + c \cdot \delta^i - c \cdot \delta \\
\sqrt{(x_1^j - x_1)^2 + (x_2^j - x_2)^2 + (x_3^j - x_3)^2} &= c \cdot \Delta t^j + c \cdot \delta^j - c \cdot \delta \\
\sqrt{(x_1^k - x_1)^2 + (x_2^k - x_2)^2 + (x_3^k - x_3)^2} &= c \cdot \Delta t^k + c \cdot \delta^k - c \cdot \delta \\
\sqrt{(x_1^m - x_1)^2 + (x_2^m - x_2)^2 + (x_3^m - x_3)^2} &= c \cdot \Delta t^m + c \cdot \delta^m - c \cdot \delta
\end{aligned}$$

Using a Linear Approximation

First Order Taylor Series About $\overline{x}$

$$\rho^j(x) \approx \rho^j(\overline{x}) - \frac{\partial \rho^j(\overline{x})}{\partial \overline{x}_1}\Delta x_1 - \frac{\partial \rho^j(\overline{x})}{\partial \overline{x}_2}\Delta x_2 - \frac{\partial \rho^j(\overline{x})}{\partial \overline{x}_3}\Delta x_3$$

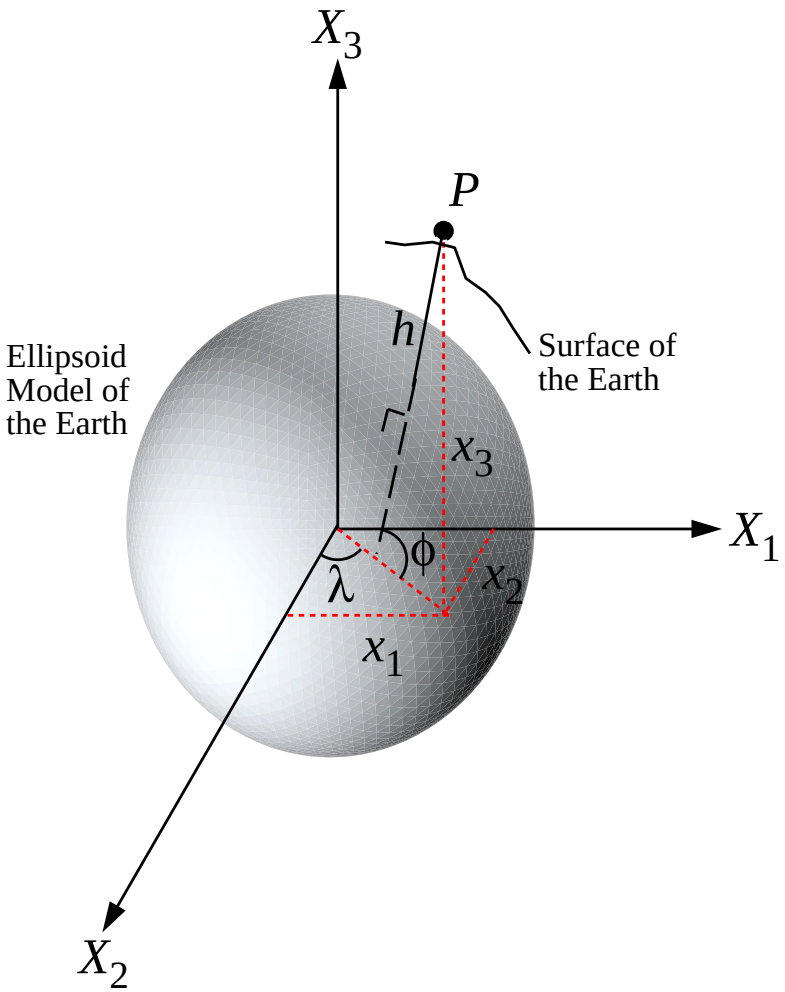
The Resulting System

$$\rho^j(\overline{x}) - \frac{x_1^j - \overline{x}_1}{\rho^j(\overline{x})}\Delta x_1 - \frac{x_2^j - \overline{x}_2}{\rho^j(\overline{x})}\Delta x_2 - \frac{x_3^j - \overline{x}_3}{\rho^j(\overline{x})}\Delta x_3 = c \cdot \Delta t^j + c\delta^i - c\delta$$

The New Unknowns

$$\Delta x_1,\ \Delta x_2,\ \Delta x_3, \text{ and } \delta$$

Ellipsoidal and Cartesian Coordinates



Obtaining the Latitude, Longitude and Altitude

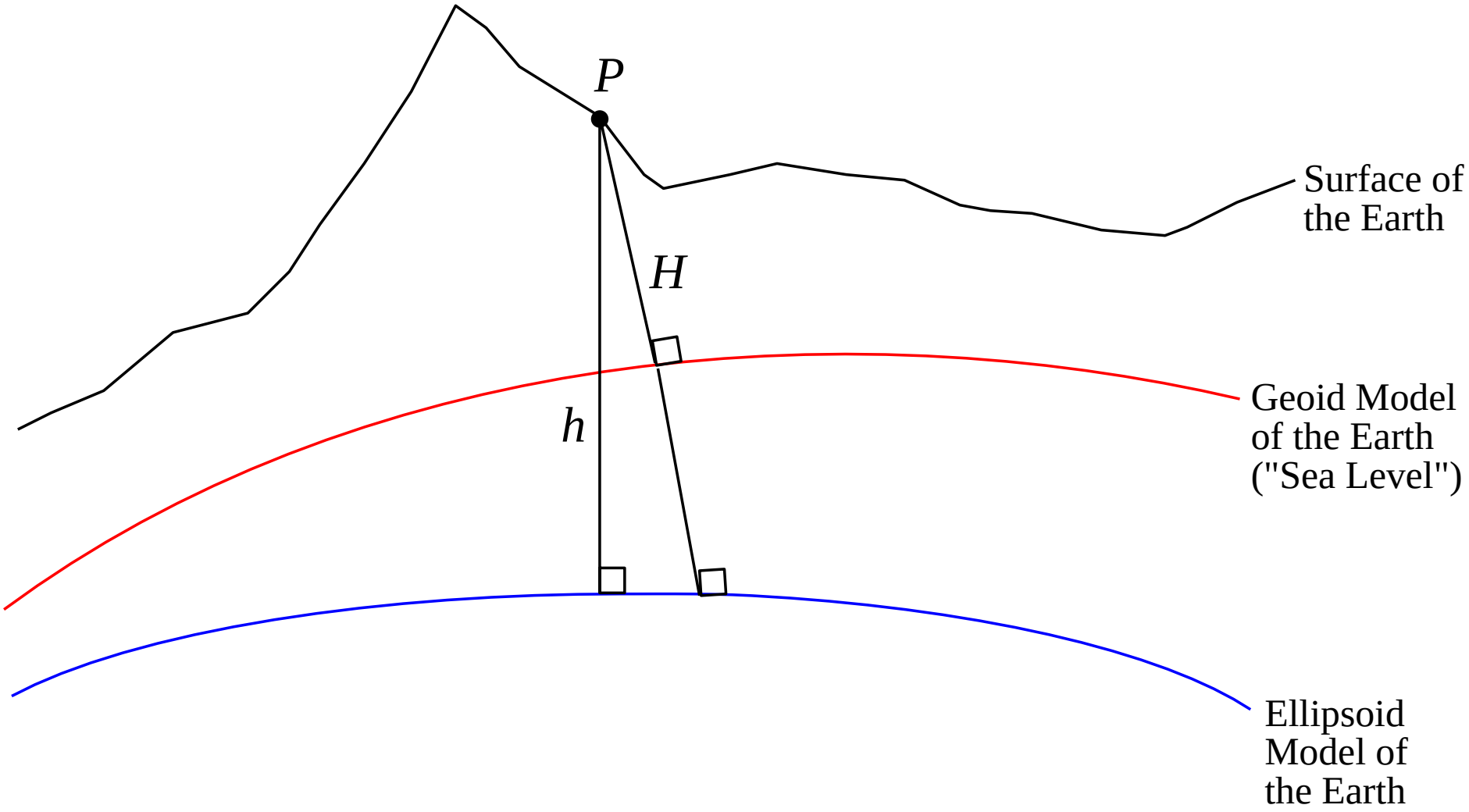
- From Ellipsoidal to Cartesian Coordinates:

$$
\begin{aligned}
x_1 &= \left(\frac{a^2}{\sqrt{a^2 \cos^2 \phi + b^2 \sin^2 \theta}} h \right) \cos \phi \cos \lambda \\
x_2 &= \left(\frac{a^2}{\sqrt{a^2 \cos^2 \phi + b^2 \sin^2 \theta}} h \right) \cos \phi \sin \lambda \\
x_3 &= \left(\frac{b^2}{a^2} \frac{a^2}{\sqrt{a^2 \cos^2 \phi + b^2 \sin^2 \theta}} h \right) \cos \phi \cos \lambda
\end{aligned}
$$

- From Cartesian to Ellipsoidal Coordinates:

  Solve for (λ, ϕ, h) given (x_1, x_2, x_3)

A Closer Look at Altitude



The Map-Matching Problem

- Notation:

  Finite set of streets, $\overline{\mathcal{N}}$

  Reporting times, $\{0, 1, \ldots, T\}$

  Actual location at time t, $\overline{P}^t$

  Estimate location at time t, P^t

- The Problem:

  Determine the street in $\overline{\mathcal{N}}$ that contains $\overline{P}^t$

The Map-Matching Problem (cont.)

- An Important Issue:

  We do not know the street system, $\overline{\mathcal{N}}$, exactly

- More Notation:

  Network representation, $\mathcal{N}$

  Each arc, $A \in \mathcal{N}$, is a curve in $\mathbb{R}^2$

The Map-Matching Problem (cont.)

- Assumptions:

  Arc $A \in \mathcal{N}$ can be completely characterized by a finite sequence of points $(A^0, A^1, \ldots, A^{n_a})$

  There is a one-to-one correspondence between the arcs in $\mathcal{N}$ and the streets in $\overline{\mathcal{N}}$

- Still More Notation:

  A^0 and A^{n_a} are referred to as nodes

  $(A^1, A^2, \ldots, A^{n_a-1})$ are referred to as shape points

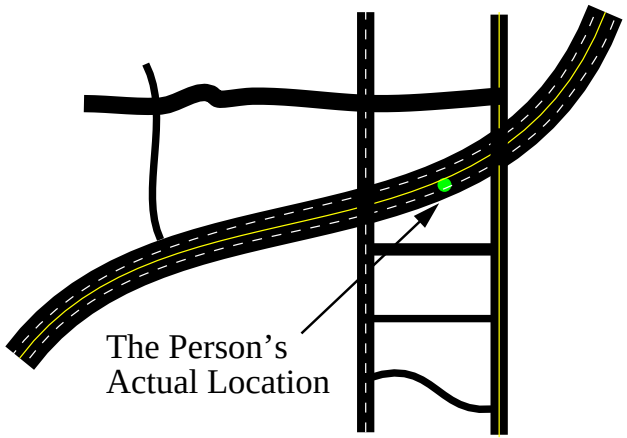
The Map-Matching Problem (cont.)

The Process

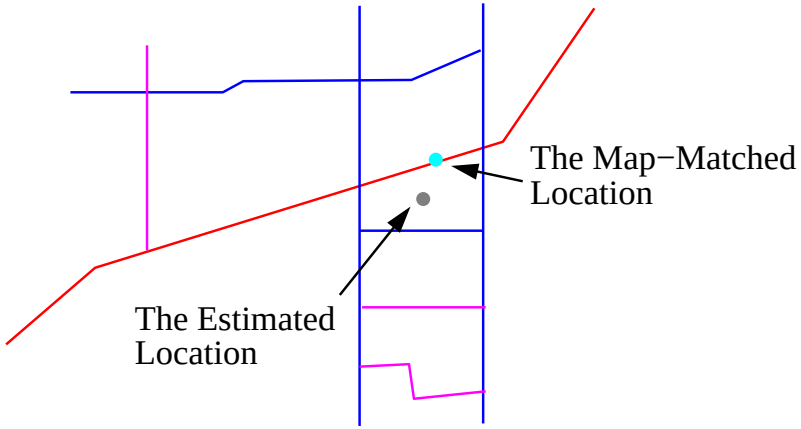
1. Match the estimated location, P^t, with an arc, A in the "map", $\mathcal{N}$
2. Then determine the street, $\overline{A} \in \overline{\mathcal{N}}$, that corresponds to the person's actual location, $\overline{P}^t$

The Map-Matching Problem (cont.)

The Set of (Actual) Streets



The Set of (Estimated) Arcs



Geometric Point-to-Point Matching

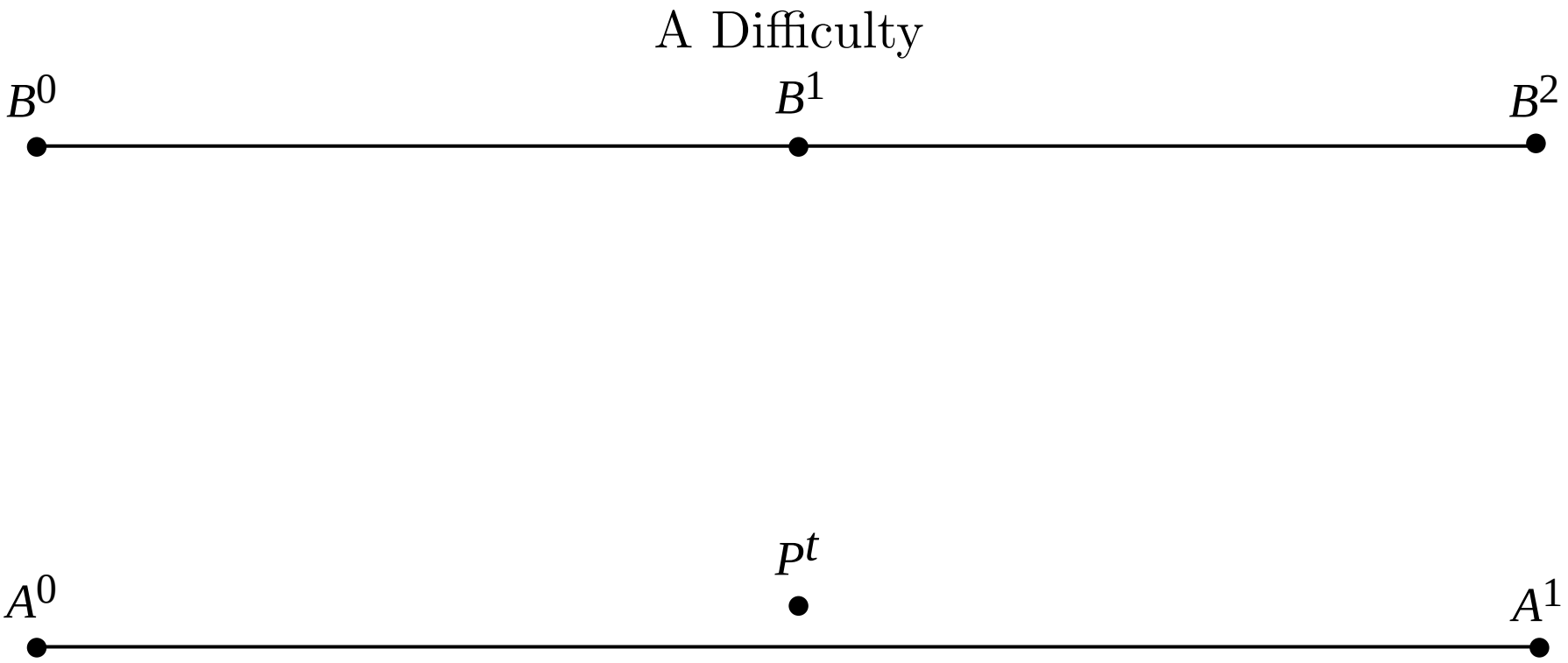
- The Idea:

  Match P^t to the "closest" node or shape point in the network

- One Approach:

  $$||x - y||_2 = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

Geometric Point-to-Point Matching (cont.)



Geometric Point-to-Curve Matching

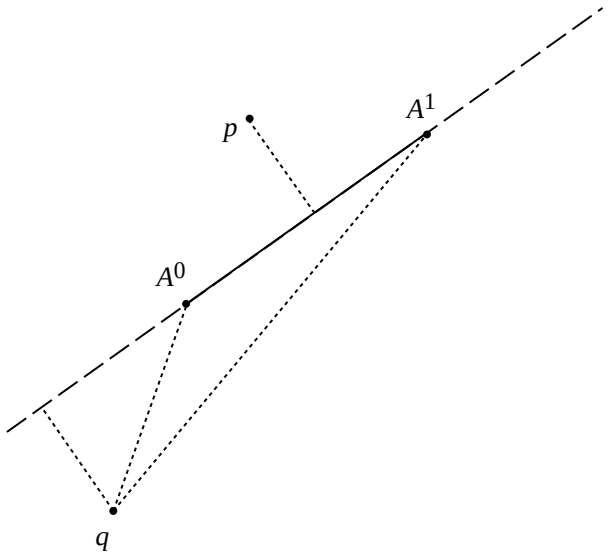
- The Idea:

  Identify the arc in $\mathcal{N}$ that is closest to P^t

- One Approach:

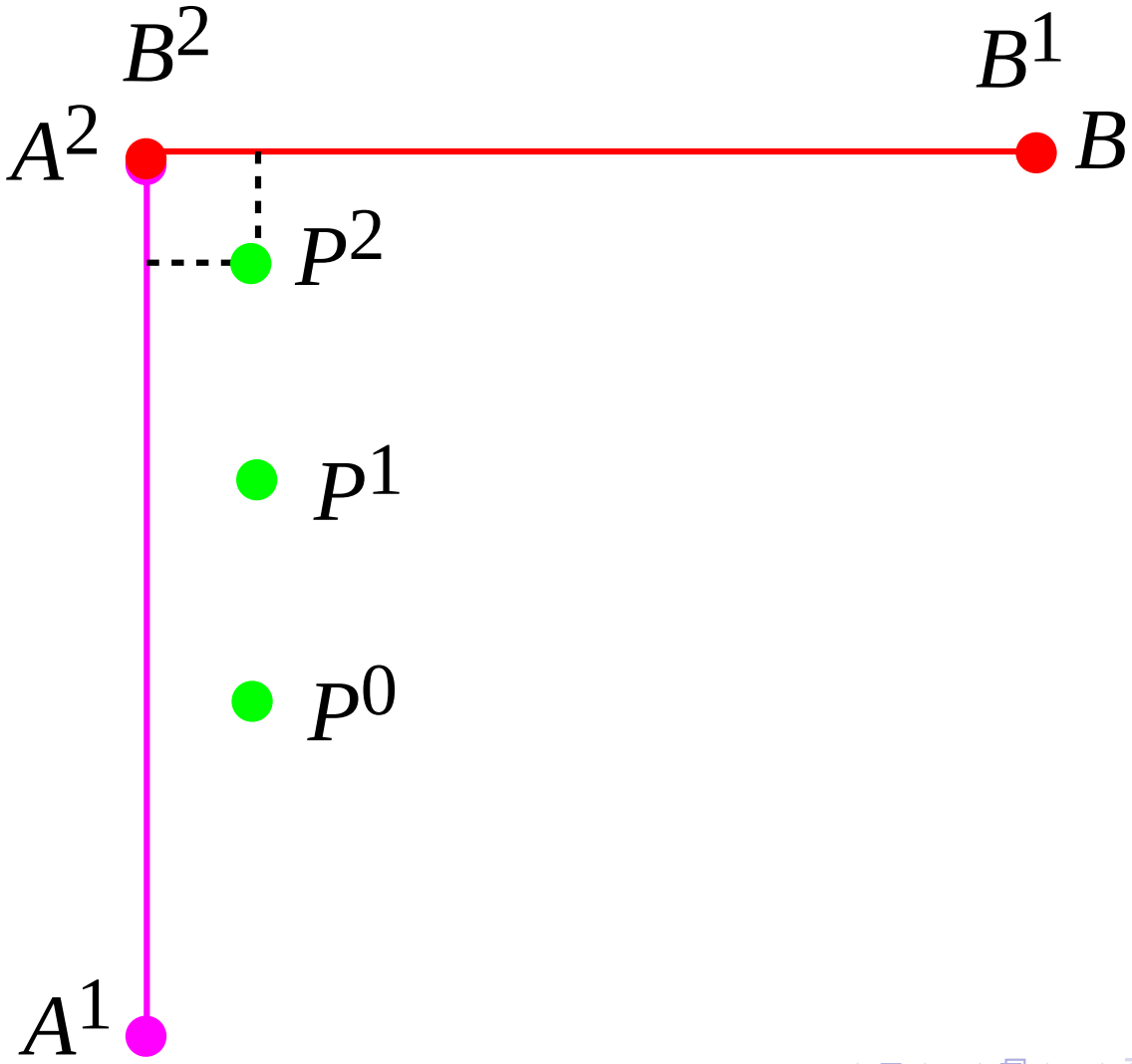
  Let $\{\lambda a + (1-\lambda)b, \lambda \in \mathbb{R}\}$ denote the line, A, through a and b

$$d(c, A) = \sqrt{\frac{[(a_2-b_2)c_1+(b_1-a_1)c_2+(a_1b_1-b_1a_2)]^2]}{(a_2-b_2)^2+(b_1-a_1)^2}}$$

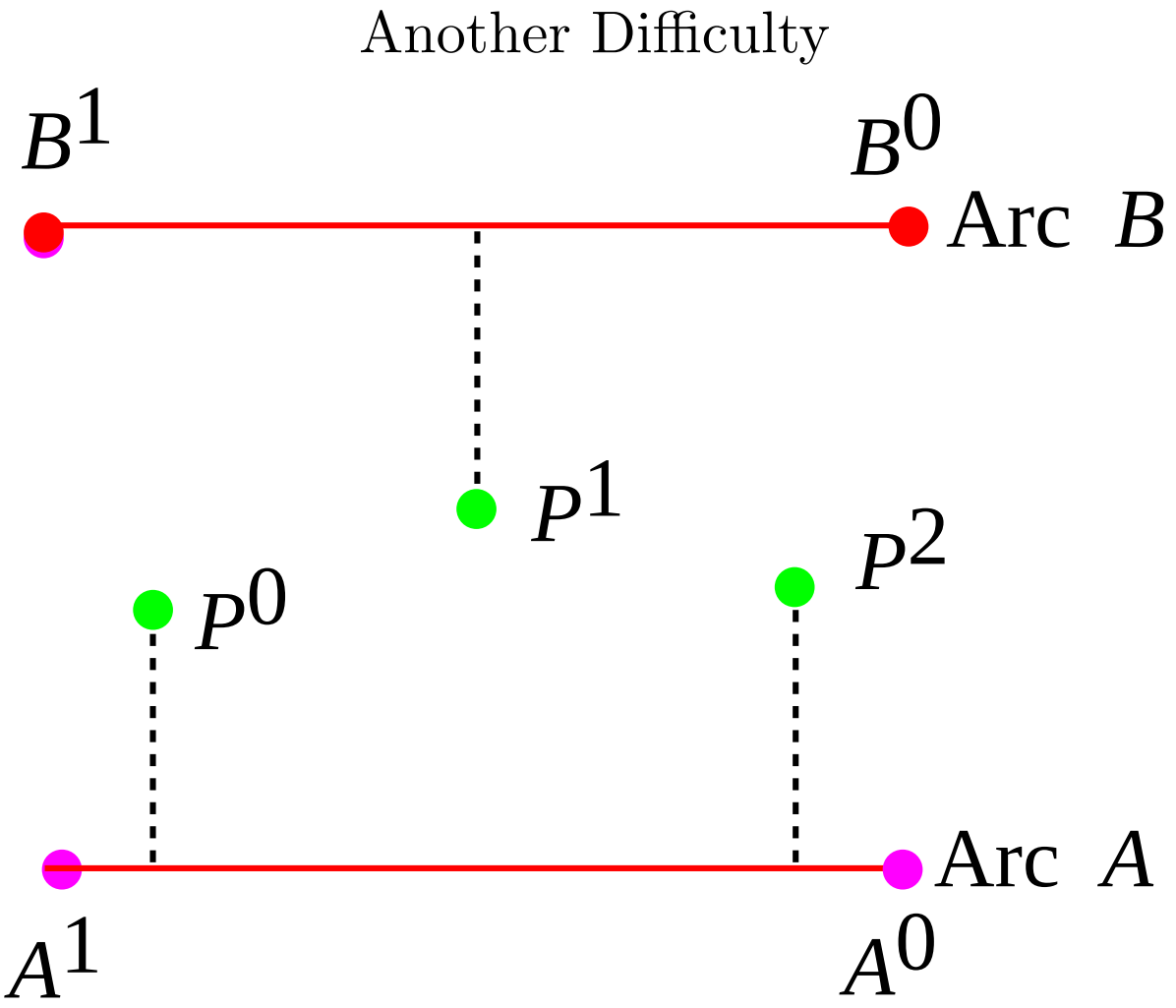


Geometric Point-to-Curve Matching (cont.)

One Difficulty



Geometric Point-to-Curve Matching (cont.)



Geometric Curve-to-Curve Matching

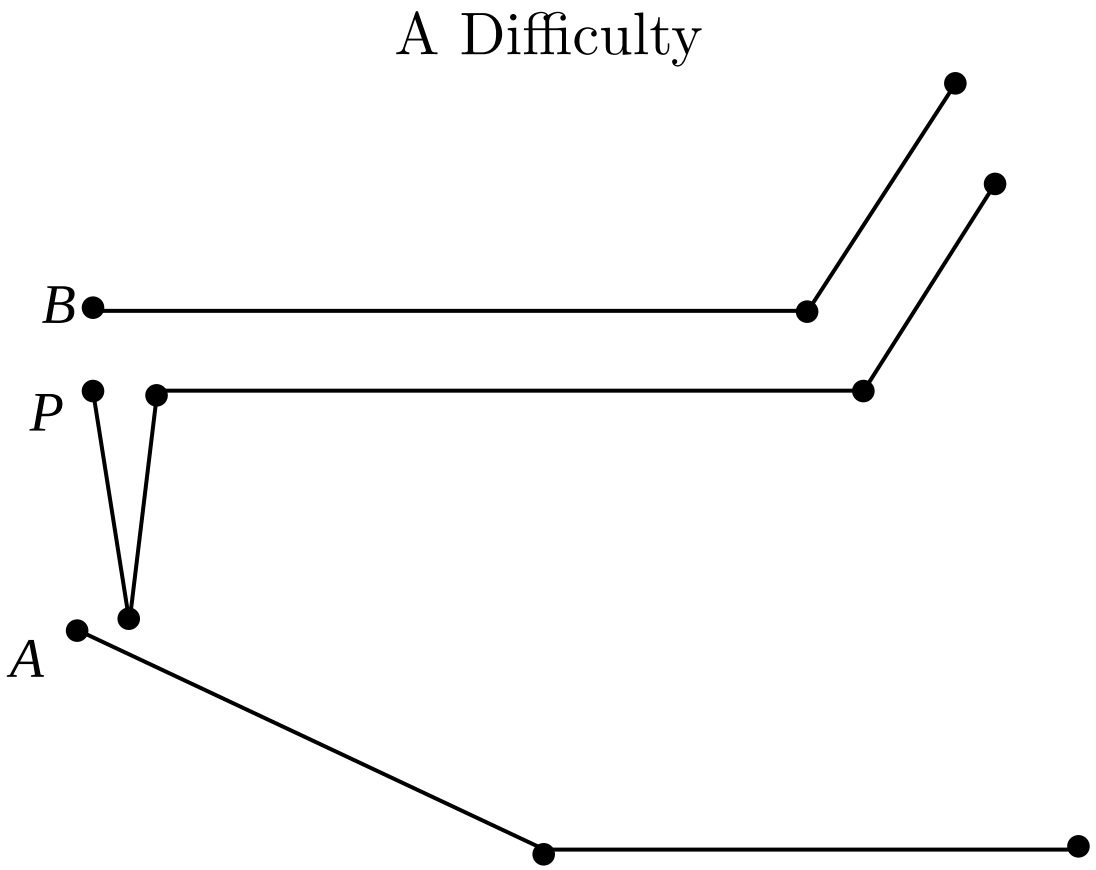
- The Idea:

  Matching to the arc that is closest to the piecewise linear curve, P defined by the points $P^0, P^1, \ldots, P^m$

- One Approach:

  $||A - B||_{\min} \min_{a \in A, b \in B} ||a - b||$

Geometric Curve-to-Curve Matching (cont.)

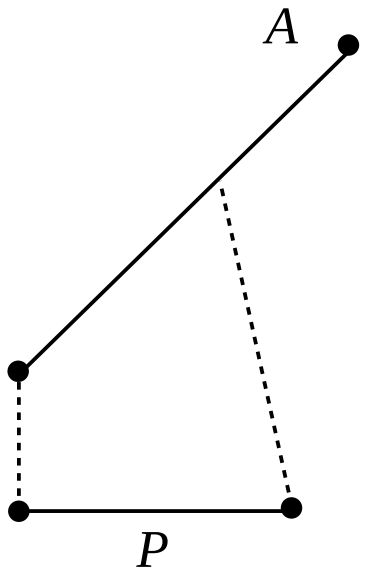


Geometric Curve-to-Curve Matching (cont.)

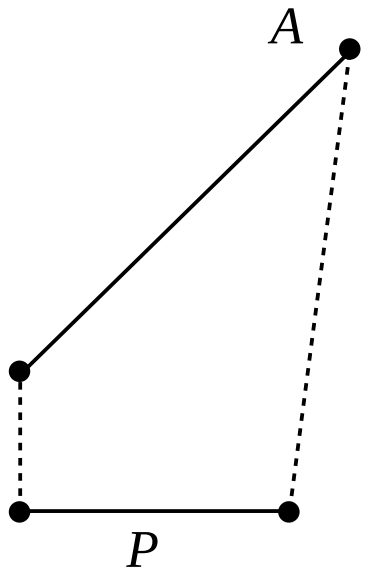
- Another Approach:

$$||A - B||_2 = \int_0^1 ||a(t) - b(t)||dt$$

- A Difficulty:



The Distance Between Equal Length Subsets of P and A



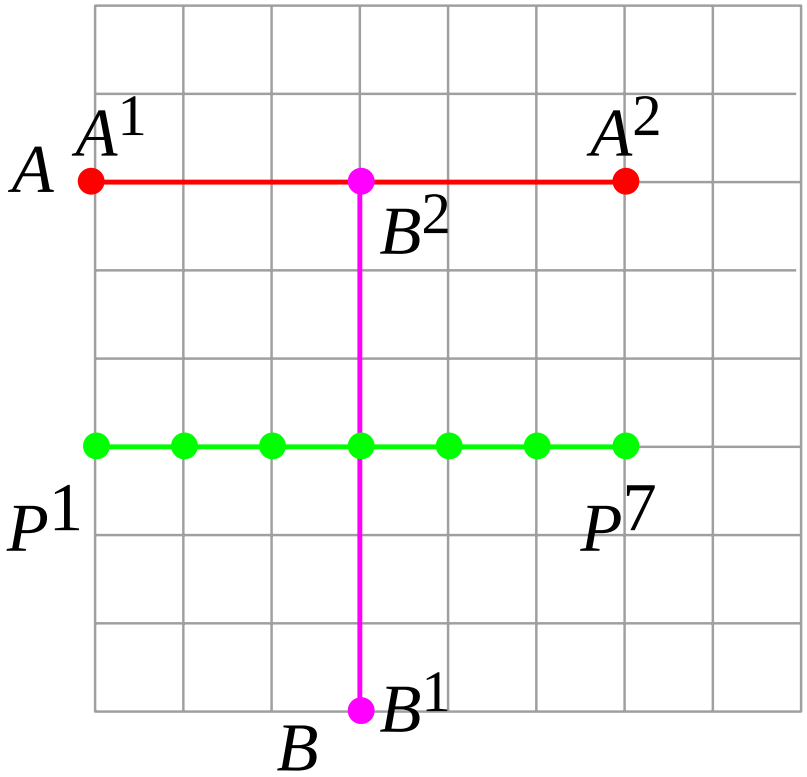
The Distance Between All of P and All of A

Geometric Curve-to-Curve Matching (cont.)

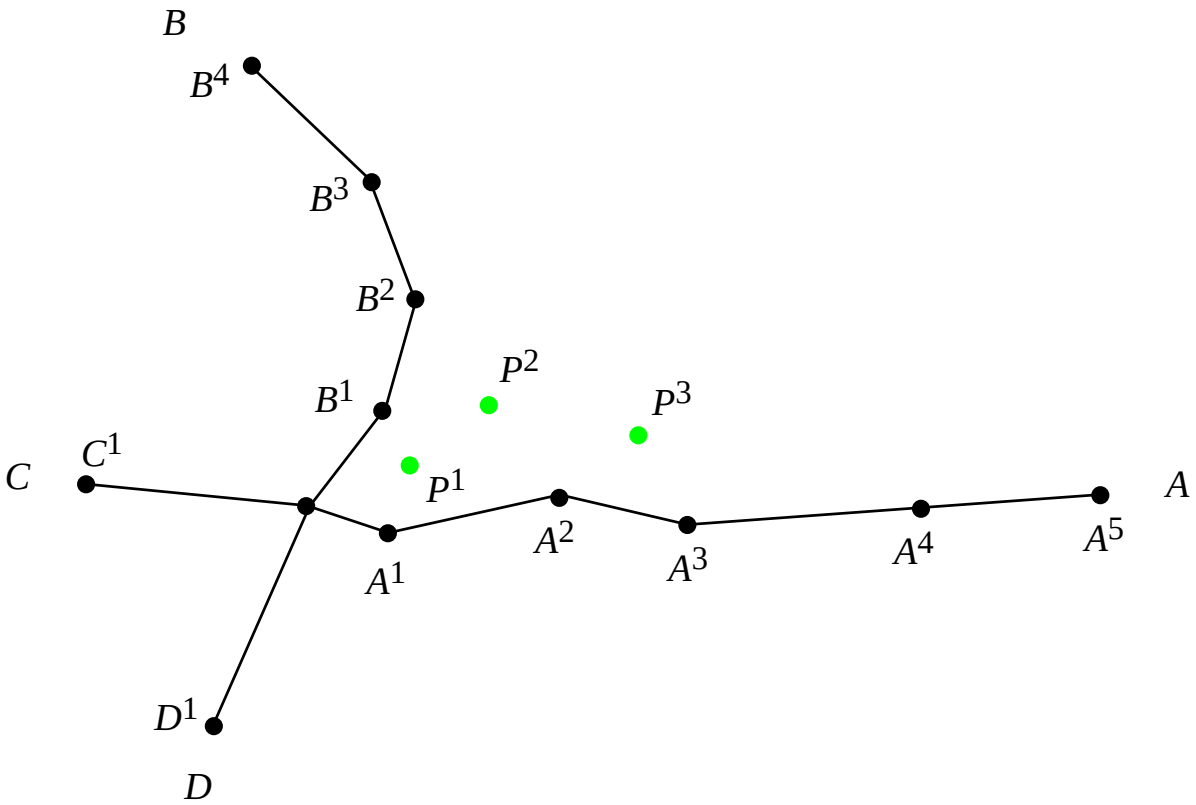
- Another Approach:

  "Correct" for differences in length

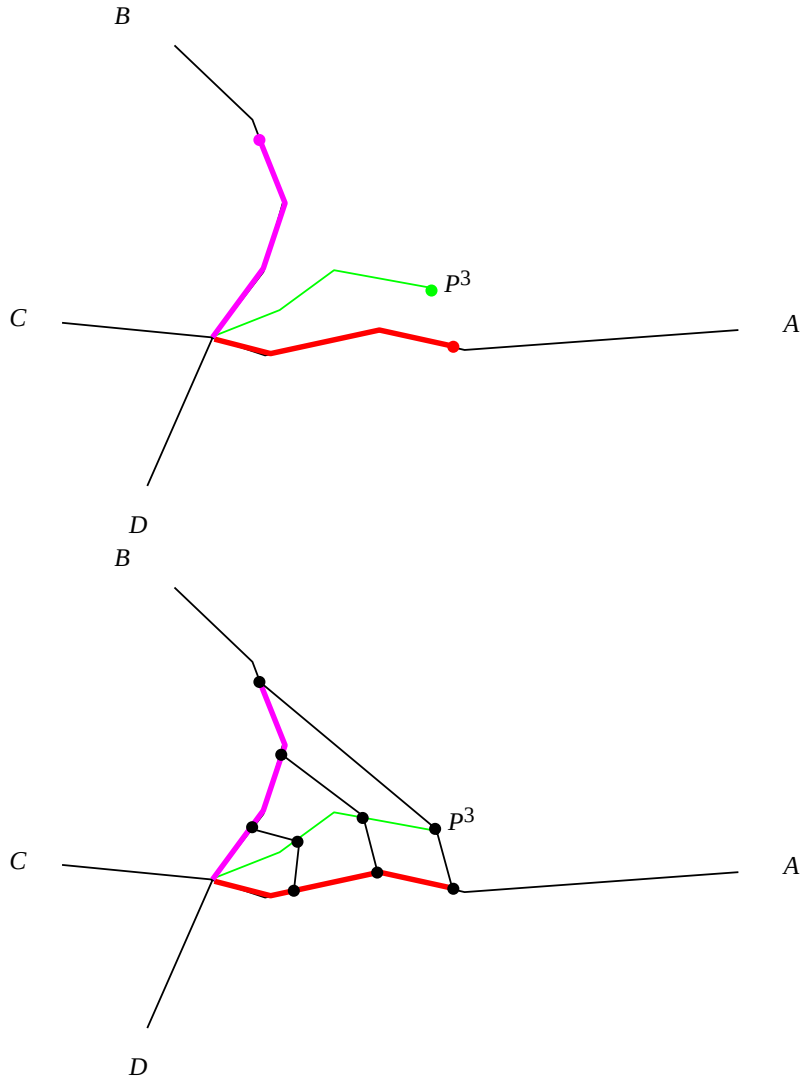
- A Difficulty:



Curve-to-Curve Matching Using Topology



Curve-to-Curve Matching Using Topology (cont.)



Route Finding

- What We Now Know:

  Where we are

- What We Need:

  How to get to the destination

An Abstract Label-Setting Algorithm

Make the origin the working node.

WHILE The working node is not the destination do the following:

> Update the temporary labels and remember the smallest
>
> Make the node with the smallest temporarily label permanent and make it the new working node.

ENDWHILE

Label-Setting Details

FOR For all nodes that are reachable from the working node:

Calculate the distance to this node through the working node.
IF This distance is less than the nodes current label:

Set the nodes label equal to this distance.
Set the nodes predecessor equal to the working node.

ENDIF
IF The new label is less than the best so far:

Remember the node and its label.

ENDIF

ENDFOR

FOR For all temporary nodes:

IF The label is less than the best so far:

Remember the node and its label.

ENDIF

ENDFOR

Worst-Case Asymptotic Efficiency: A First Look

Number of Top-Level Iterations:

Each iteration "completes" one node
m nodes in the network

Work per Top-Level Iteration:

Update each node's label (m of them)
Keep track of smallest

Overall Complexity:

m^2

The Collection of Reachable Nodes

Updating Labels:

We know that only labels of nodes that are reachable from the working node can change.

Each node can have an associated array of pointers to reachable nodes.

Each node can only be updated as many times as there are inbound arcs.

So, total work for updates equals the number of arcs.

Finding the Smallest Temporary Label:

We can "sort" the temporary nodes based on their labels.

Since we only ever want the smallest label, a good data structure to use is a heap.

Worst-Case Asymptotic Efficiency: Revisited

Number of Top-Level Iterations:

- Each iteration "completes" one node
- m nodes in the network
- Total work for updates is based on number of arcs (n)

Work per Top-Level Iteration:

- Adjust the heap ($\log m$)
- Find the best node and remove it from the heap ($\log m$)

Overall Complexity:

- $m \log m + n$

What to do When You're Off-Route

- "Brute Force" Re-Calculation
- Sensitivity Analysis
- Pre-Calculation

Other Interesting Routing Problems

- Finding an Alternative Route
- Incorporating Time and Tolls

Alternatives to the Best Route

- Common Approaches:
  - Link elimination
  - Find the r-best route
- Unfortunately:
  - Link elimination can lead to infeasibilities
  - r-best routes are often similar to the best
  - Neither captures what users really want

Alternatives to the Best Route (cont.)

- Another Approaches:

  A route p is k-similar to a route s if p and s have at most k links in common

- Some More Notation:

  A denotes the *node-arc incidence matrix* which has components a_{ij} defined as follows: $a_{ij} = 1$ if link j is directed out of node i, $a_{ij} = -1$ if link j is directed into node i, and $a_{ij} = 0$ otherwise

  The vector b denotes the origin and destination (i.e., $b_O = 1$, $b_D = -1$, and $b_i = 0, i \in \mathcal{N} - \{O, D\}$)

  The vector c denotes link "costs"

Alternatives to the Best Route (cont.)

An Optimization Formulation of the Traditional Problem

$$\begin{array}{rrcl} \min_x & c^\top x & & \\ \text{s.t.} & Ax & = & b \\ & x & \geq & 0 \end{array} \tag{1}$$

An Optimization Formulation of the k-Similar Problem

$$\begin{array}{rrcl} \min_x & c^\top x & & \\ \text{s.t.} & Ax & = & b \\ & z^\top x & \leq & k \\ & x & \in & \{0,1\}^n \end{array} \tag{2}$$

Alternatives to the Best Route (cont.)

Using Lagrangian Relaxation

$$\begin{array}{lrcl} \max_{\lambda} & \min_{x} & c^{\top}x + \lambda(z^{\top}x - k) & \\ & \text{s.t.} & Ax & = & b \\ & & x & \geq & 0 \end{array} \tag{3}$$

Incorporating Nonadditive Costs

- Nonlinear Valuation of Time:
  Component costs include time and money

  Small amounts of time have relatively low value whereas large amounts of time are very valuable

- Nonadditive Tolls and Fares:
  The tolls/fare on a path is almost never the sum of the tolls/fares on the links in that path

  Tolls on the NJ Turnpike:

		To				
		Exit 1	Exit 2	Exit 3	Exit 4	Exit 5
	Exit 1		0.45	0.70	0.95	1.20
	Exit 2	0.45		0.45	0.60	0.85
From	Exit 3	0.70	0.45		0.35	0.50
	Exit 4	0.95	0.60	0.35		0.35
	Exit 5	1.20	0.85	0.50	0.35	

Incorporating Nonadditive Costs (cont.)

Even More Notation:

t is the vector of times
τ is the vector of tolls
v is the value of time

An Optimization Formulation of the "Time and Tolls" Problem

$$\begin{array}{lrcl} \min_x & v(t^\top x) + \tau^\top x & & \\ \text{s.t.} & Ax & = & b \\ & x & \in & \{0,1\}^n. \end{array} \tag{4}$$

Motivating an Algorithm

Suppose we know the solution, x^*. Then, we could instead "solve":

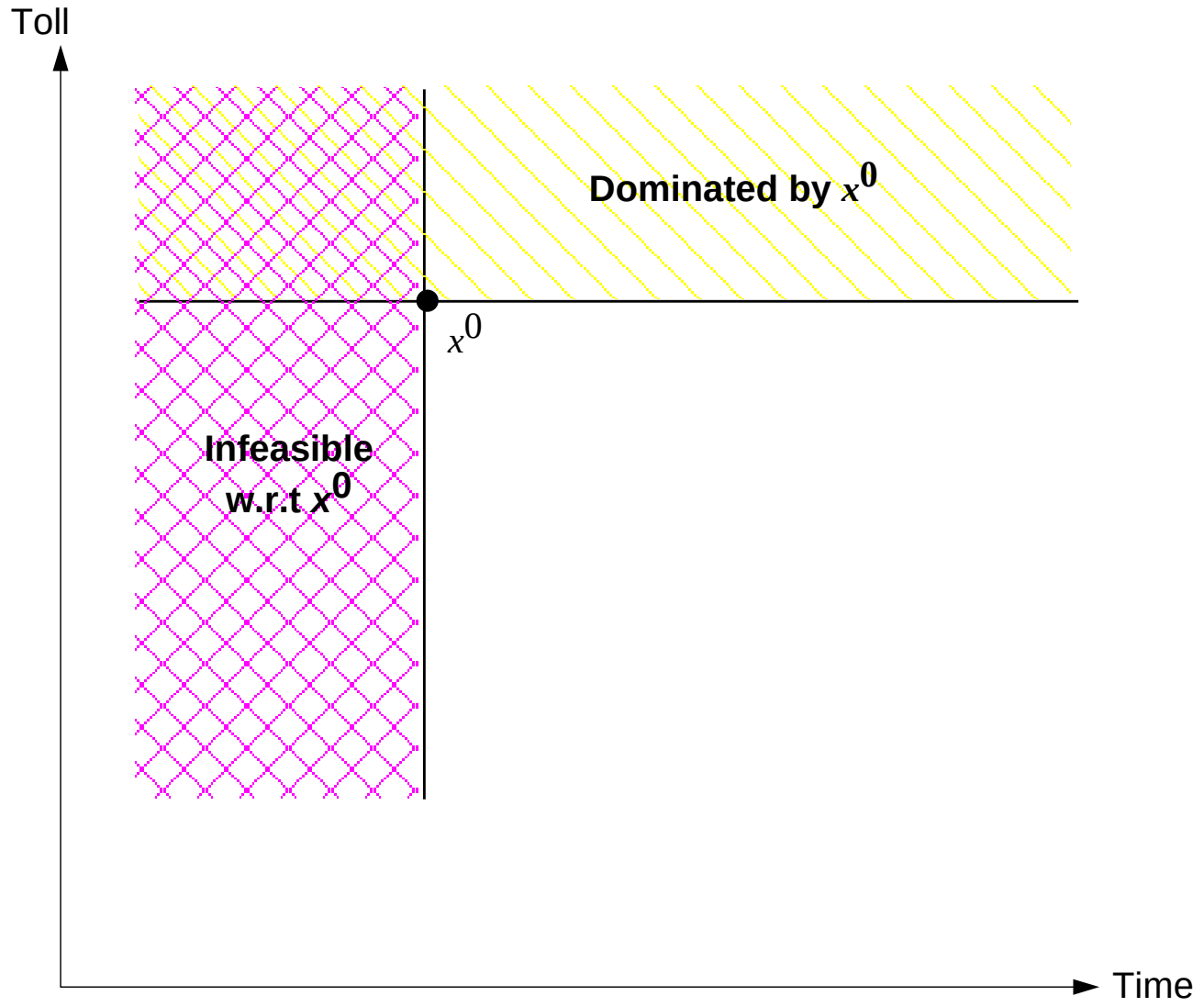
$$\begin{array}{rrcl} \min_x & v\left(t^\top x\right) & & \\ \text{s.t.} & Ax & = & b \\ & \tau^\top x & = & \tau^\top x^* \\ & x & \in & \{0,1\}^n. \end{array} \tag{5}$$

If v is monotone, then this is equivalent to:

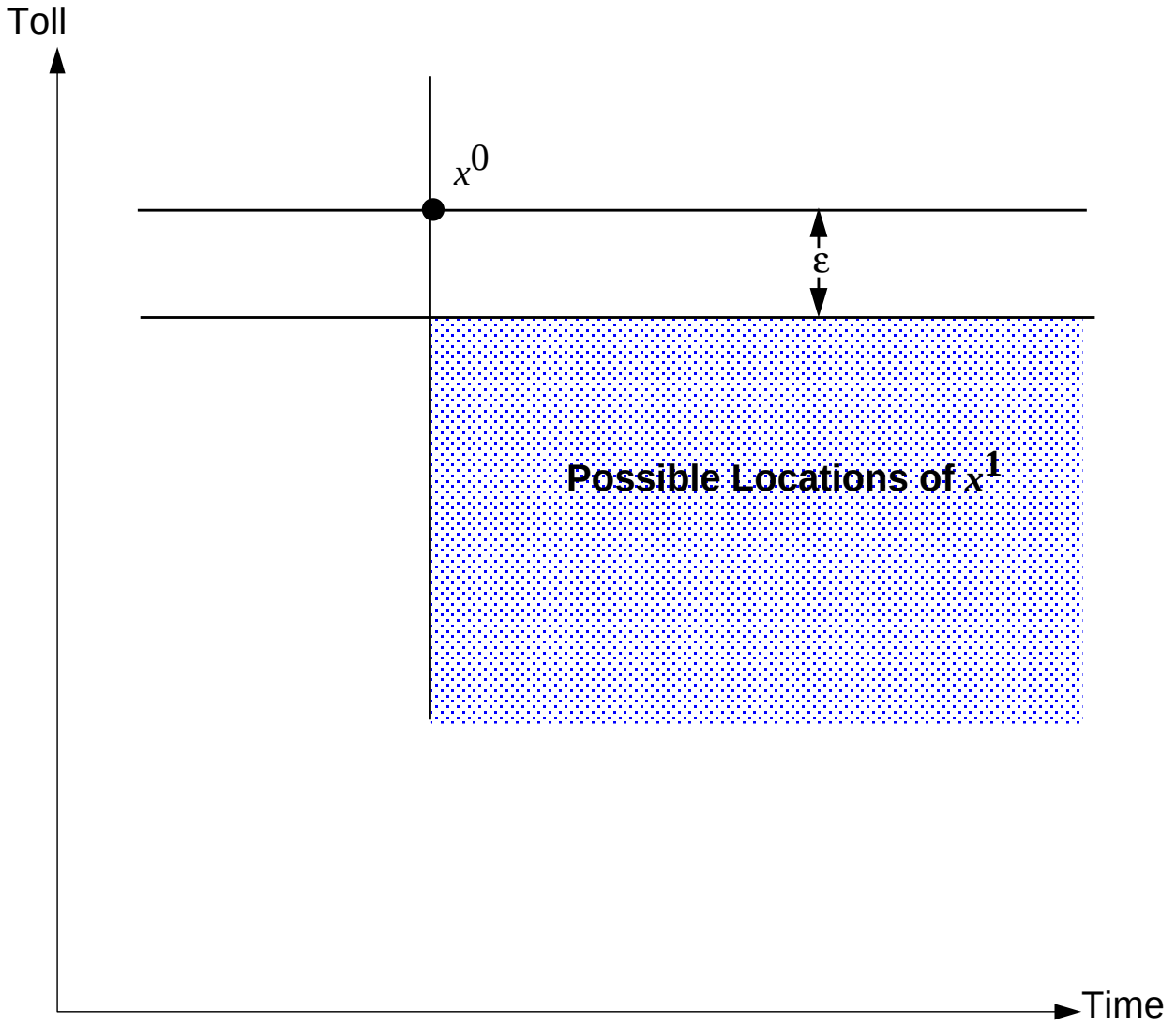
$$\begin{array}{rrcl} \min_x & t^\top x & & \\ \text{s.t.} & Ax & = & b \\ & \tau^\top x & = & \tau^\top x^* \\ & x & \in & \{0,1\}^n \end{array} \tag{6}$$

What we do is "guess" at x^* and then solve this constrained shortest path problem

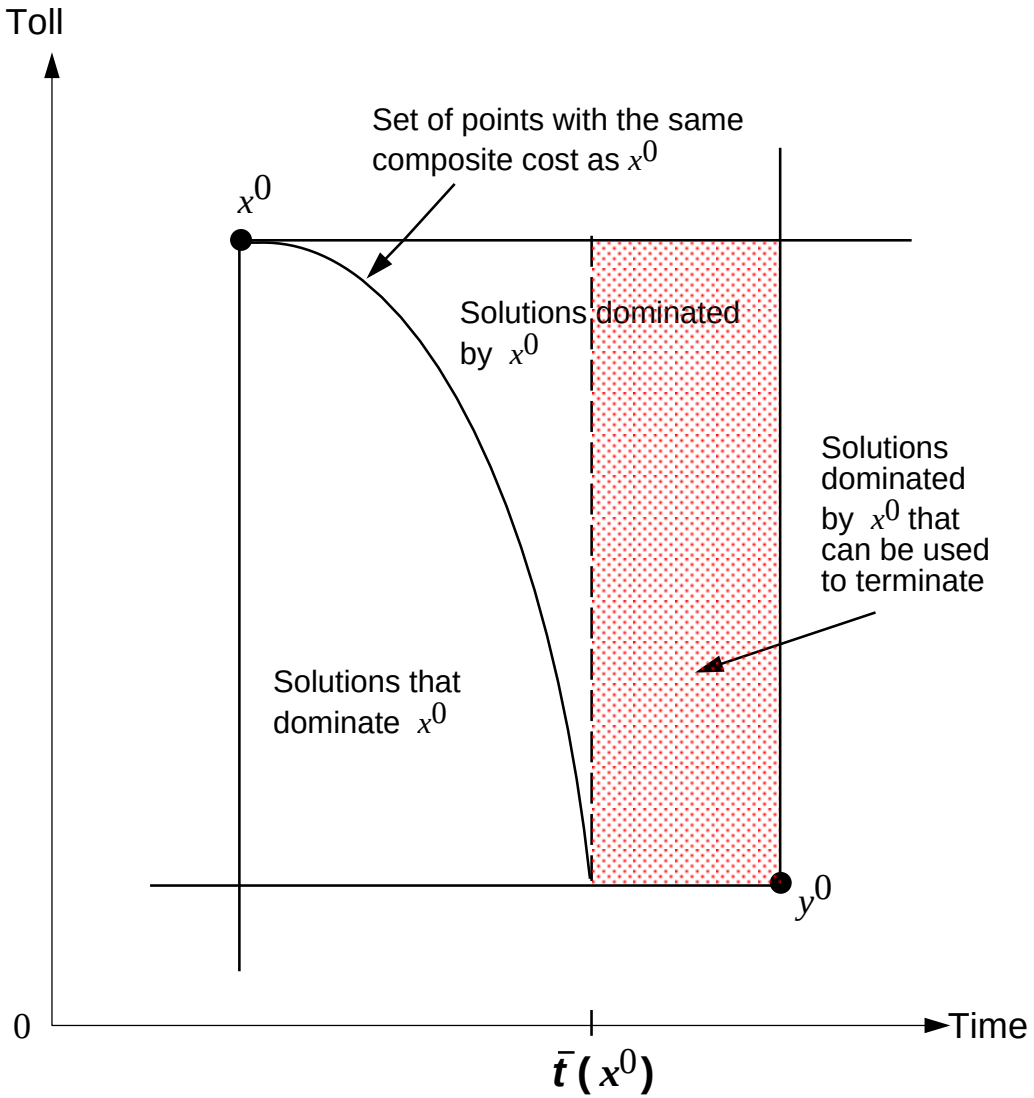
Visualization of the Algorithm



Visualization of the Algorithm - Forcing a New Solution



Improving the Performance of the Algorithm



Conclusions

- From the topics I covered today:

  Research in mathematics has been important in the development of PNS

  Research in computer science has been important in the development of PNS

- From other things I've done:

  Research in software engineering has been important in the development of PNS

Next Generation Systems

- Real-Time Traffic Requires Research On:
  - Combining data from multiple sources
  - Use of incomplete and stochastic data
  - Use of time-dependent data
  - Traffic forecasting
- Hardware Changes Require Research On:
  - Effective use of multiple cores
  - Multi-modal trips (e.g., driver+transit+walk)
  - Coordination of multiple vehicles (e.g., “Where should we meet?”)