On–line Calibration for Dynamic Traffic Assignment

Constantinos Antoniou

October 5, 2007

Seminar at Portland State University
Outline

• Introduction

• Formulation

• Solution approaches

• Case study

• Directions for further research

• A glimpse at ongoing research
DynaMIT

DYnamic Network Assignment for the Management of Information to Travelers
DynaMIT is a simulation-based real-time system predicting traffic providing travel information
Simulation

**Demand**
- Micro simulator
- Departure time choice
- Route choice
- Response to ITS
- Origin-destination flows

**Supply**
- Mesoscopic Traffic simulator
Real-Time

- Continuous data collection
- Fast processing
  - Past: Originally designed as distributed software
  - Present: “monolithic”
  - Future: parallelization efforts underway
    - Processors no longer getting faster, but adding cores
- Immediate dissemination
Traffic prediction

- Rolling horizon
- Consistency
Rolling Horizon

DynaMIT constantly predicts traffic conditions for a pre-specified time period
Broadcasting forecasts does not affect the weather

But with traffic…
DynaMIT Evaluation

MITSIM

Measures of effectiveness

Surveillance System

Information

DynaMIT
DynaMIT: Network Representation
DynaMIT

Overall Framework

Information dissemination

Database
Network representation
Historical information

Real-Time inputs
Traffic Surveillance and Control

State Estimation

Demand Simulation

Supply Simulation

Prediction-based Information Generation

Demand Simulation

Supply Simulation

Information Generation

Information dissemination
Motivation

• Models/components
  – Demand–side
    * OD flows
    * Behavioral models (e.g. route choice)
  – Supply–side
    * Speed–density relationship (e.g. $u = u_f \left[ 1 - \left( \frac{\max(0, K - K_{min})}{K_{jam}} \right)^\beta \right]^\alpha$
      - where $u$ denotes the speed, $u_f$ is the free flow speed, $K$ is the density, $K_{min}$ is the minimum density, $K_{jam}$ is the jam density and $\alpha$ and $\beta$ are model parameters.)
    * Segment capacities

• Existing state estimation and model calibration approaches include subset of these models
Literature review

• Demand parameters
  – OD estimation [Ashok and Ben–Akiva (1993, 2000)]

• Supply parameters
  – Capacity estimation [van Arem and van der Vlist (1992)]
Research objectives

• Develop an integrated approach for on–line calibration that
  – Jointly estimates and predicts demand and supply parameters
  – Is general and flexible
    * Not tied to a particular DTA system
    * Applicable to any calibration parameters
    * Can exploit any available information
  – Is computationally feasible

• Demonstrate the feasibility of the approach
Inputs and outputs

- Historical data
- A priori parameter values
- On-line calibration
- Surveillance data
- On-line calibrated parameters
Formulation

• State–space model
  – State vector
  – Measurement equations
  – Transition equations

• Idea of deviations
**State vector**

- Deviations $\Delta \pi_h$ of model inputs and parameters $\pi_h$ from available estimates

\[
\Delta \pi_h = \pi_h - \pi_h^H
\]

- State vector includes
  - OD flows $\chi_{ijh}$: number of vehicles departing from origin $i$ towards destination $j$ during time interval $h$
  - Model parameters, e.g.
    * route choice model parameters
    * speed–density relationship parameters
    * segment capacities
Measurement equations (1)

- Direct measurement equation

\[ \pi_h^a = \pi_h + v_h \]

- In deviations’ form (subtracting \( \pi_h^H \) from both sides)

\[
\begin{align*}
\pi_h^a - \pi_h^H &= \pi_h - \pi_h^H + v_h \\
\Delta \pi_h^a &= \Delta \pi_h + v_h
\end{align*}
\]

where

- \( a \) denotes \textit{a priori} values, \( H \) indicates historical estimates, \( v_h \) is a random error vector.
Measurement equations (II)

• Indirect measurement equation

\[ M_h = S(\pi_h) + \nu_h \]

• In deviations’ form (subtracting \( M_h^H \) from both sides)

\[ M_h - M_h^H = S(\pi_h) - M_h^H + \nu_h \Rightarrow \]
\[ \Delta M_h = S(\pi_h^H + \Delta \pi_h) - M_h^H + \nu_h \]

where

– \( M_h \) is a vector of measurements for interval \( h \), and \( \nu_h \) is a random error vector.
– \( S \) is the simulator function mapping the state to the measurements (no analytic expression).
Transition equations

\[ \pi_{h+1} = \sum_{q=h-p}^{h} F_{q}^{h+1} \cdot \pi_{q} + \eta_{h} \]

- In deviations’ form (subtracting \( \pi^{H} \) for appropriate interval from both sides)

\[ \pi_{h+1} - \pi_{h+1}^{H} = \sum_{q=h-p}^{h} F_{q}^{h+1} (\pi_{q} - \pi_{q}^{H}) + \eta_{h} \Rightarrow \]

\[ \Delta \pi_{h+1} = \sum_{q=h-p}^{h} F_{q}^{h+1} \cdot \Delta \pi_{q} + \eta_{h} \]

where

- \( F_{q}^{h+1} \) is a transition matrix of effects of \( \Delta \pi_{q} \) on \( \Delta \pi_{h+1} \)
- \( p \) is the degree of the autoregressive process, and
- \( \eta_{h} \) is a random error vector
The model at a glance

\[ \Delta \pi_h = \Delta \pi_h + \nu_h \]

\[ \Delta M_h = S(\pi_h^H + \Delta \pi_h) - M_h^H + \nu_h \]

\[ \Delta \pi_{h+1} = \sum_{q=h-p}^{h} \mathcal{F}_{q}^{h+1} \cdot \Delta \pi_q + \eta_h \]
Solution approaches

• Linear models
  – Kalman Filter (KF)

• Non-linear models
  – Extended Kalman Filter (EKF)
  – Limiting Extended Kalman Filter (LimEKF)
  – Unscented Kalman Filter (UKF)
Kalman Filtering principles

• Prediction–correction approach

1. **Predict** *a priori* state for interval $h$ (using information up to $h - 1$)

2. Compute *Kalman gain* matrix

3. Combine Kalman gain and surveillance from interval $h$ to **correct** state prediction for interval $h$
Extended Kalman Filter

- Extension of Kalman Filter for non-linear models
  - First order Taylor expansion approximation

- Can use numerical derivatives, if no analytical relationship exists (for measurement equations)
  - Large number of function evaluations is required
  - e.g. \( 2n \) evaluations if central differences are used (\( n \) is state dimension)

- Computation of Kalman gain most “expensive” operation
Limiting Extended Kalman Filter

- Use a pre–computed Kalman gain
  - E.g.: (weighted) average of Kalman gains (computed off–line)
  - No need to compute the Kalman gain on–line

- Only a single function evaluation is required on–line

- Can run EKF off–line and periodically re–compute Kalman gain
Unscented Kalman Filter

• Uses *Unscented Transformation*
  – The state mean and covariance are used to generate $2n + 1$ *sigma points* ($n$ is the dimension of the state)
  – These points are propagated through the *true* nonlinearity

• Kalman gain is computed from the state and measurement covariances

• No *limiting* version is possible
  – *Mean* measurement vector (from $2n + 1$ replications) is required for the *correction* stage
Case study — Objectives

• Demonstrate feasibility of the on–line calibration approach

• Verify importance of on–line calibration
  – Impact of joint estimation and prediction of demand and supply parameters

• Test candidate algorithms based on several criteria
  – Performance/Fit (estimation and prediction)
  – Computational properties
  – Robustness
The DynaMIT–R system

• State–of–the–art, simulation–based DTA system
  – Transportation network and supply characteristics
  – OD demand and travel behavior

• Operates in rolling horizon for:
  – Real–time estimation of network performance
  – Short–term prediction of future network conditions
  – Generation of traffic information

• Inputs
  – A-priori OD flows
  – Route choice parameters
  – Traffic dynamics models’ parameters
  – Segment capacities
Network (II)
Southampton, U.K. (35km portion of freeway M27)

- 45 segments (three types)
- 10 sensors
- 20 OD pairs
- PM data (peak direction)
Traffic flow characteristics

Sensor counts (all days)

Counts (veh/hr/lane)

15:00 16:00 17:00 18:00 19:00

Speeds/Densities (dry days)

Density (veh/km/lane)

Speed (km/h)

15:00 16:00 17:00 18:00 19:00

Speeds/Densities (wet days)

Density (veh/km/lane)

Speed (km/h)

15:00 16:00 17:00 18:00 19:00
Experimental design

• Type of day: Dry/wet weather

• Scope of on–line calibration: demand only vs. joint demand and supply

<table>
<thead>
<tr>
<th>Scope</th>
<th>Dry</th>
<th>Wet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demand only (base)</td>
<td>KF/GLS</td>
<td>KF/GLS</td>
</tr>
<tr>
<td></td>
<td>EKF</td>
<td>EKF</td>
</tr>
<tr>
<td>Demand and supply</td>
<td>LimEKF</td>
<td>LimEKF</td>
</tr>
<tr>
<td></td>
<td>UKF</td>
<td>UKF</td>
</tr>
</tbody>
</table>
Measures of effectiveness

1. Fit of estimated/predicted vs. observed speeds

2. Fit of estimated/predicted vs. observed sensor counts
   • Aggregate statistic: Normalized root mean square error (RMSN)

\[
\text{RMSN} = \sqrt{N \times \sum_{n=1:N} (y_n - \hat{y}_n)^2} / \sum_{n=1:N} y_n
\]

3. Computational performance
Summary results for estimated and predicted speeds (Dry)

![Graph showing RMSN for different prediction methods: Estimation, One-step Prediction, Two-step Prediction, Three-step Prediction. The methods include Base, EKF, LimEKF, and UKF.](graph.png)
Summary results for estimated and predicted speeds (Wet)

![Graph showing RMSN for different predictions](image)
On–line calibrated speed–density relationships
Mainline segments – EKF

top: dry weather, bottom: wet weather
Computational performance (I)

- EKF/UKF
  - Similar performance ($2n$ vs. $2n + 1$ function evaluations)

- Limiting Kalman Filter
  - Order-of-magnitude improvement
  - Single function evaluation required (independent of state dimension)
### Computational performance (II)

<table>
<thead>
<tr>
<th>Function</th>
<th>Function evaluations</th>
<th>Runtime (min:sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EKF</td>
<td>2n</td>
<td>98:29</td>
</tr>
<tr>
<td>LimEKF</td>
<td>1</td>
<td>0:43</td>
</tr>
<tr>
<td>UKF</td>
<td>2n</td>
<td>97:44</td>
</tr>
</tbody>
</table>

- Pentium 4 2.8GHz computer with 512MB RAM
Additional analysis - LimEKF
(top: speeds, bottom: counts)

- RMSN
- Estimation
- One-step pred.
- Two-step pred.
- Three-step pred.
- Four-step pred.
- Five-step pred.
Conclusions/Findings (I)

- Joint on-line calibration of demand and supply parameters can improve prediction accuracy

- Changes in parameters are consistent with traffic engineering principles and expectations

- Interactions between many parameters are captured
  - Many degrees of freedom
Conclusions/Findings (II)

- EKF/UKF: “Real-time” performance in a small, but non-trivial real network

- Limiting EKF
  - Order(s) of magnitude improvement in computational performance
  - Comparable accuracy to EKF/UKF
  - Robustness
Conclusions/Findings (III)

- EKF has more desirable properties than UKF (in this application)
  - EKF generally outperforms UKF
    * Full power of UKF is not exhibited, because transition equation is already linear
  - UKF is not as robust (perhaps due to the number of points used)
  - EKF and UKF have similar computational properties
  - Limiting EKF (while no similar UKF version exists)
Directions for further research

- Further experimental analysis
  - Scalability
  - Robustness
  - Sensitivity

- Impact of grouping segments (for speed–density relationship estimation)

- Algorithms
  - e.g. particle filters (generalization of UKF)
A glimpse at ongoing research
Mesoscopic traffic simulation

Typical approach

Density

\[ u = f(k) \]

Calculate speed

Advance vehicles

Alternative approach

Density and other explanatory variables

Database

Calculate speed

Advance vehicles
More flexible functional forms

Classic vs. loess speed–density relationship

Plus: easily incorporate additional measurements
A comparison of data-driven approaches (I)

- Locally weighted regression
- Support vector regression
- Neural networks
A comparison of data–driven approaches (II)

<table>
<thead>
<tr>
<th></th>
<th>RMSPE</th>
<th>RMSN</th>
<th>MPE</th>
<th>U</th>
<th>$U^m$</th>
<th>$U^s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typical (reference case)</td>
<td>0.04802</td>
<td>0.04641</td>
<td>-0.01685</td>
<td>0.02314</td>
<td>0.15077</td>
<td>0.01773</td>
</tr>
<tr>
<td>Loess (density)</td>
<td>0.04382</td>
<td>0.04094</td>
<td>-0.00266</td>
<td>0.02027</td>
<td>0.00841</td>
<td>0.00519</td>
</tr>
<tr>
<td>% improved</td>
<td>-9%</td>
<td>-12%</td>
<td>-84%</td>
<td>-12%</td>
<td>-94%</td>
<td>-71%</td>
</tr>
<tr>
<td>NN (density)</td>
<td>0.04535</td>
<td>0.04173</td>
<td>-0.00562</td>
<td>0.02068</td>
<td>0.02210</td>
<td>0.00019</td>
</tr>
<tr>
<td>% improved</td>
<td>-6%</td>
<td>-10%</td>
<td>-67%</td>
<td>-11%</td>
<td>-85%</td>
<td>-99%</td>
</tr>
<tr>
<td>SVR (density)</td>
<td>0.04550</td>
<td>0.04202</td>
<td>-0.00362</td>
<td>0.02080</td>
<td>0.00953</td>
<td>0.00035</td>
</tr>
<tr>
<td>% improved</td>
<td>-5%</td>
<td>-9%</td>
<td>-79%</td>
<td>-10%</td>
<td>-94%</td>
<td>-98%</td>
</tr>
<tr>
<td>Loess (density+lagflow)</td>
<td>0.03747</td>
<td>0.03578</td>
<td>-0.00542</td>
<td>0.01773</td>
<td>0.02611</td>
<td>0.00001</td>
</tr>
<tr>
<td>% improved</td>
<td>-22%</td>
<td>-23%</td>
<td>-68%</td>
<td>-23%</td>
<td>-83%</td>
<td>-100%</td>
</tr>
<tr>
<td>NN (density+lagflow)</td>
<td>0.04116</td>
<td>0.03756</td>
<td>-0.00513</td>
<td>0.01861</td>
<td>0.02515</td>
<td>0.00065</td>
</tr>
<tr>
<td>% improved</td>
<td>-14%</td>
<td>-19%</td>
<td>-70%</td>
<td>-20%</td>
<td>-83%</td>
<td>-96%</td>
</tr>
<tr>
<td>SVR (density+lagflow)</td>
<td>0.03839</td>
<td>0.03635</td>
<td>-0.00610</td>
<td>0.01801</td>
<td>0.03036</td>
<td>0.00013</td>
</tr>
<tr>
<td>% improved</td>
<td>-20%</td>
<td>-22%</td>
<td>-64%</td>
<td>-22%</td>
<td>-80%</td>
<td>-99%</td>
</tr>
</tbody>
</table>

* improved refers to improvement over the offline (reference case)
A comparison of data–driven approaches (III)

Notes: Application runtimes. SVR is an order of magnitude slower.
An integrated machine learning approach

Training data (Speeds/Densities/flows) → k-means, Fuzzy c-means, etc → Clustering

Flexible relationships f(x) → Fitting

Explanatory data (Densities/flows) → K-nearest neighbors → Classification

Flexible relationships f(x) → Estimation

Estimated speeds
An integrated approach: loess and clustering (I)

B. Distribution of speed deviations by approach

![Box plot showing distribution of speed deviations by approach](image)
An integrated approach: loess and clustering (II)

Reduction in RMSN:
Loess (density): -7%, Loess (density+flow): -35%
Loess/clust (density): -41%, Loess/clust (density+flow): -47%
An integrated approach: loess and clustering (III)

<table>
<thead>
<tr>
<th>Approach</th>
<th>Computational cost (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training</td>
</tr>
<tr>
<td>Classic speed-density</td>
<td>N/A</td>
</tr>
<tr>
<td>Loess (density)</td>
<td>0.44</td>
</tr>
<tr>
<td>Loess (density+flow)</td>
<td>1.72</td>
</tr>
<tr>
<td>Loess-cluster (density)</td>
<td>1.40</td>
</tr>
<tr>
<td>Loess-cluster (density+flow)</td>
<td>1.88</td>
</tr>
</tbody>
</table>

N/A: not applicable
Thank you

Questions?

Contact information:
Constantinos Antoniou (antoniou@central.ntua.gr)