Cadyts – a free calibration tool for dynamic traffic simulations

Gunnar Flötteröd[1]

September 10, 2009

[1] many thanks to Yu Chen for helping with the simulations

Cadyts – calibration of dynamic traffic simulations

- objective: demand calibration from traffic counts

- classical perspective: estimation of path flows
- this work: estimation of micro-simulated behavior

Outline

Classical perspective

Disaggregate (“multi-agent”) perspective

Applications and technicalities

Example result

Outline

Classical perspective

Disaggregate (“multi-agent”) perspective

Applications and technicalities

Example result

Classical perspective: model entities

$n = 1 \dots N$	origin/destination (OD) pairs
d_n	number of trips between OD pair n
C_n	set of available routes for OD pair n
d_{ni}	number of trips on route $i \in C_n$

- total demand levels are fixed

$$d_n = \sum_{i \in C_n} d_{ni} \tag{1}$$

- variable demand levels $\longrightarrow$ add fictitious routes that bypass the physical network

Classical perspective: assignment

- path flows $\mathbf{d} = (d_{ni})$ are in stochastic user equilibrium (SUE) if

$$d_{ni} = P_n(i|\mathbf{d})d_n \quad \forall n, i \in C_n \tag{2}$$

  where $P_n(i|\mathbf{d})$ is the congestion-dependent route choice model
- equivalent formulation: $\mathbf{d}$ maximizes the prior entropy

$$W(\mathbf{d}) = \prod_{n=1}^{N} d_n! \frac{\prod_{i \in C_n} (P_n(i|\mathbf{d}))^{d_{ni}}}{\prod_{i \in C_n} d_{ni}!} \tag{3}$$

- represents prior knowledge of analyst

Classical perspective: calibration

- observed traffic counts $\mathbf{y}$ depend on real path flows $\mathbf{d}$ through likelihood $p(\mathbf{y}|\mathbf{d})$
- combine this information with prior knowledge about SUE route flows
- Bayesian approach: maximize posterior entropy

$$W(\mathbf{d}|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{d}) W(\mathbf{d}) \tag{4}$$

Classical perspective: posterior path flows

- posterior entropy $W(\mathbf{d}|\mathbf{y})$ is maximized by

$$P_n(i|\mathbf{d},\mathbf{y}) = \frac{\exp(\Lambda_{ni}+\Gamma_{ni})P_n(i|\mathbf{d})}{\sum_{j\in C_n}\exp(\Lambda_{nj}+\Gamma_{nj})P_n(j|\mathbf{d})} \quad (5)$$

where

$$\Lambda_{ni} = \frac{\partial \ln p(\mathbf{y}|\mathbf{d})}{\partial d_{ni}} \quad (6)$$

$$\Gamma_{ni} = \sum_{m=1}^{N}\sum_{j\in C_m}\frac{d_{mj}}{P_m(j|\mathbf{d})}\frac{\partial P_m(j|\mathbf{d})}{\partial d_{ni}}. \quad (7)$$

- calibration problem is solved by scaling the route flow distributions

Outline

Classical perspective

Disaggregate ("multi-agent") perspective

Applications and technicalities

Example result

Disaggregate perspective: model entities

$n = 1 \dots N$	individual travelers
d_n	number of repeated choice situations
C_n	set of available travel plans for individual n
d_{ni}	number of times traveler n chooses plan $i \in C_n$
$P_n(i\|\mathbf{d})$	congestion-dependent plan choice distribution

- one choice per choice situation and traveler

$$d_n = \sum_{i \in C_n} d_{ni} \tag{8}$$

- variable demand levels $\rightarrow$ allow for stay-at-home plan

Disaggregate perspective: assignment

- micro-simulation iterates between demand and supply model
  - demand model: every traveler selects a plan
  - supply model: execute all plans in the network
- process stabilizes at stochastic fixed point under mild conditions
- average solution approximately coincides with aggregate SUE
  $\longrightarrow$ prior entropy $W(\mathbf{d})$ is still maximized

Disaggregate perspective: calibration and solution

- nothing new: combine traffic counts $\mathbf{y}$ with prior knowledge about the demand
- Bayesian approach: maximize posterior entropy ... by

$$P_n(i|\mathbf{d},\mathbf{y}) = \frac{\exp(\Lambda_{ni}+\Gamma_{ni})P_n(i|\mathbf{d})}{\sum_{j\in C_n}\exp(\Lambda_{nj}+\Gamma_{nj})P_n(j|\mathbf{d})} \tag{9}$$

  where Λ_{ni} and Γ_{ni} as before
- calibration problem is solved by scaling the individual choice distributions

Outline

Classical perspective

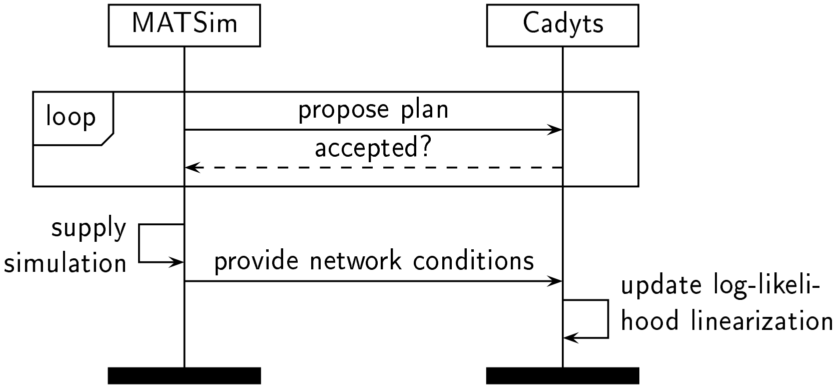
Disaggregate ("multi-agent") perspective

Applications and technicalities

Example result

Calibration of MATSim

msc one calibration iteration

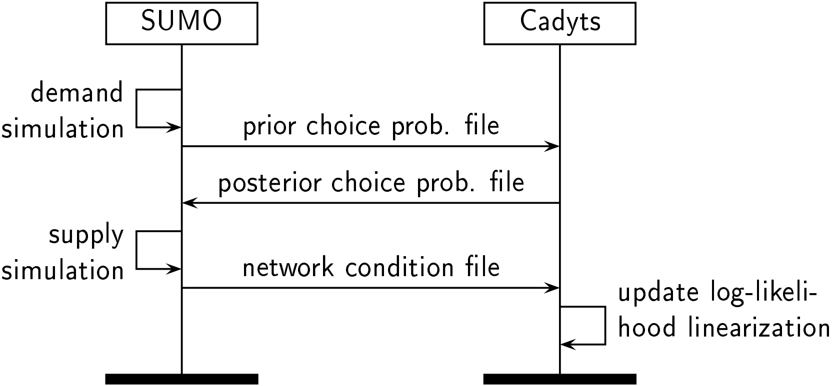


Calibration of MATSim – function calls

- MATSim and Cadyts are Java programs $\rightarrow$ function calls
- initialize the calibration:
  `void addMeasurement(L link, int start_s, int end_s, double value, Measurement.TYPE type)`
- propose plan for acceptance:
  `boolean getSampler(Object agent).isAccepted(Plan<L> plan)`
- pass network conditions to calibration:
  `void afterNetworkLoading(SimResults<L> simResults)`

Calibration of SUMO

msc one calibration iteration



Calibration of SUMO – calls to executable

- SUMO's Python code calls Cadyts jar file:
  `java -jar SumoCalibration.jar ...`
- initialize the calibration
  `... INIT -measfile meas.xml`
- ask calibration to select plans (trips):
  `... CHOICE -choicesetfile choicesets.xml -choicefile choices.xml`
- pass network conditions to calibration:
  `... UPDATE -netfile flows.xml`

Outline

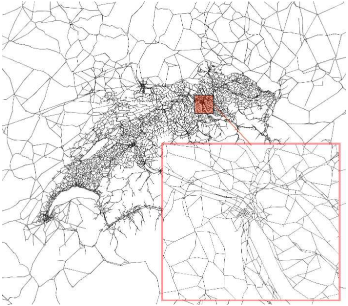
Classical perspective

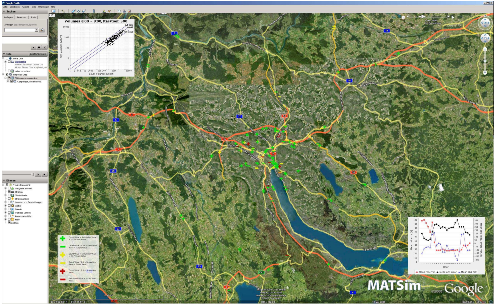
Disaggregate ("multi-agent") perspective

Applications and technicalities

Example result

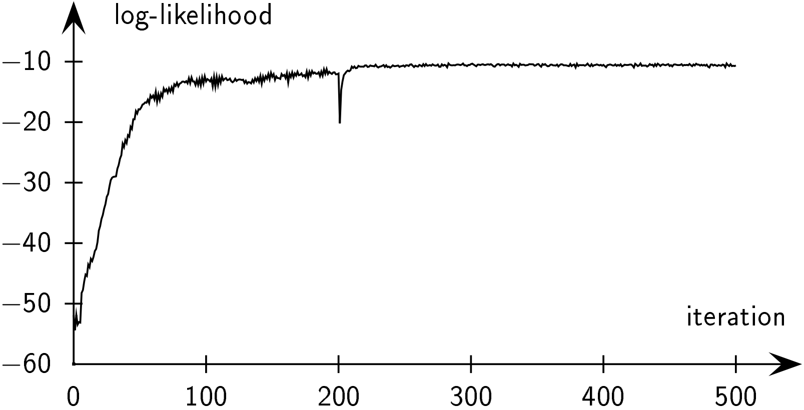
Zurich test case with MATSim





- Zurich network with 60 492 links, population of size 187 484
- calibrate route/dpt.time/mode choice from 159 inductive loops

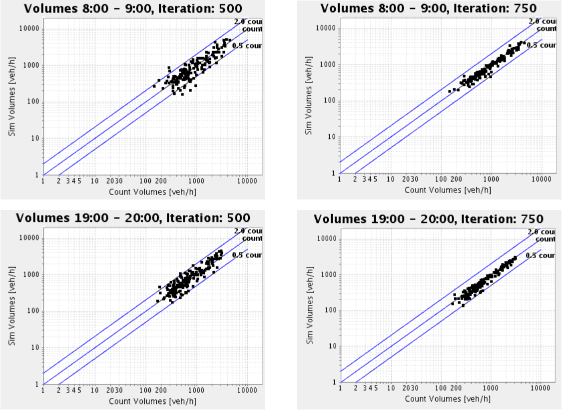
Convergence



Measurement fit

plain simulation | with calibration

morning | evening



Summary

- Cadyts calibrates disaggregate demand from traffic counts
- broad conceptual and technical applicability
- freely available → `http://transp-or2.epfl.ch/cadyts/`
- current and future work
  - generalize to calibration of demand parameters
  - incorporate further types of sensor data
  - apply to more simulators, current work: DynaMIT