Potentials of computing science in transportation: Research tool and applications

Kai Nagel

Department of Computer Science, ETH Zurich

CH-8092 Zurich, Switzerland

http://sim.inf.ethz.ch/

Foreword

There are four talks at STRC which give the technical results on which many of my remarks are based:

- Charypar, **Activity scheduling** using a genetic algorithm. Yesterday.
- Raney, **Iterative activity and route planning** for agent-based transportation simulation. This morning.
- Cetin, A large-scale multi-agent **traffic microsimulation** based on queue model. This afternoon.
- Gloor, **Modular distributed** multi-agent simulations and hiking in the Alps. Tomorrow morning.

This talk: more general issues.

Outline

- What is the **problem**? (Transportation planning)
- **Micro-simulation** of travel behavior (physical simulation vs strategy generation)
- **Physical simulation**
- **Strategy generation**
- Dynamics of the **learning system**
- **Computation**
- One **validation** result (all of Switzerland)
- Summary

What is the problem?

Questions of transportation planning

- Predict situation in 20 years from now
- Detailed analysis (e.g.: How do poor people benefit? Where do emissions go?)
- Influence of ITS (Intelligent Transportation Systems)

Note: Methods for operations similar to methods for planning.
Cf. weather vs. climate forecasting.

Traditional method: 4-step process

E.g. EMME/2, VISUM, POLYDROM.

- **Trip generation**. Determine sources and sinks.
- **Trip distribution**. Connect sources to sinks.
- **Mode choice**.
- **Route assignment**.

4-step process, ctd

Major advantage of 4-step process:

Route assignment has unique solution

(in terms of link volumes; under some conditions).

This means: *Any* correct computation will yield same result.

Simplifies analysis enormously.

4-step process, ctd

minor shortcoming of 4-step process:

Traveler behavior not coupled to demographics.

E.g.: Choice between car and train indep. of income, season ticket ownership, or car ownership (!).

This could in principle be changed.

4-step process, ctd

MAJOR shortcoming of 4-step process:

No dependence on time-of-day.

E.g.:

- No evaluation of time-dependent ITS capabilities.
- No peak-hour spreading; no scheduling reaction at all.
- In general: Use of behavioral rules not possib./plausib.
- Computation of emissions difficult to impossible.

Not known how to change this within 4-step without losing main advantage (uniqueness).

Micro-simulation

Alternative to 4-step process: Try **micro-simulation**.

Micro-simulation: *Everything* (travelers, vehicles, traffic lights, etc.) can be individually resolved ...

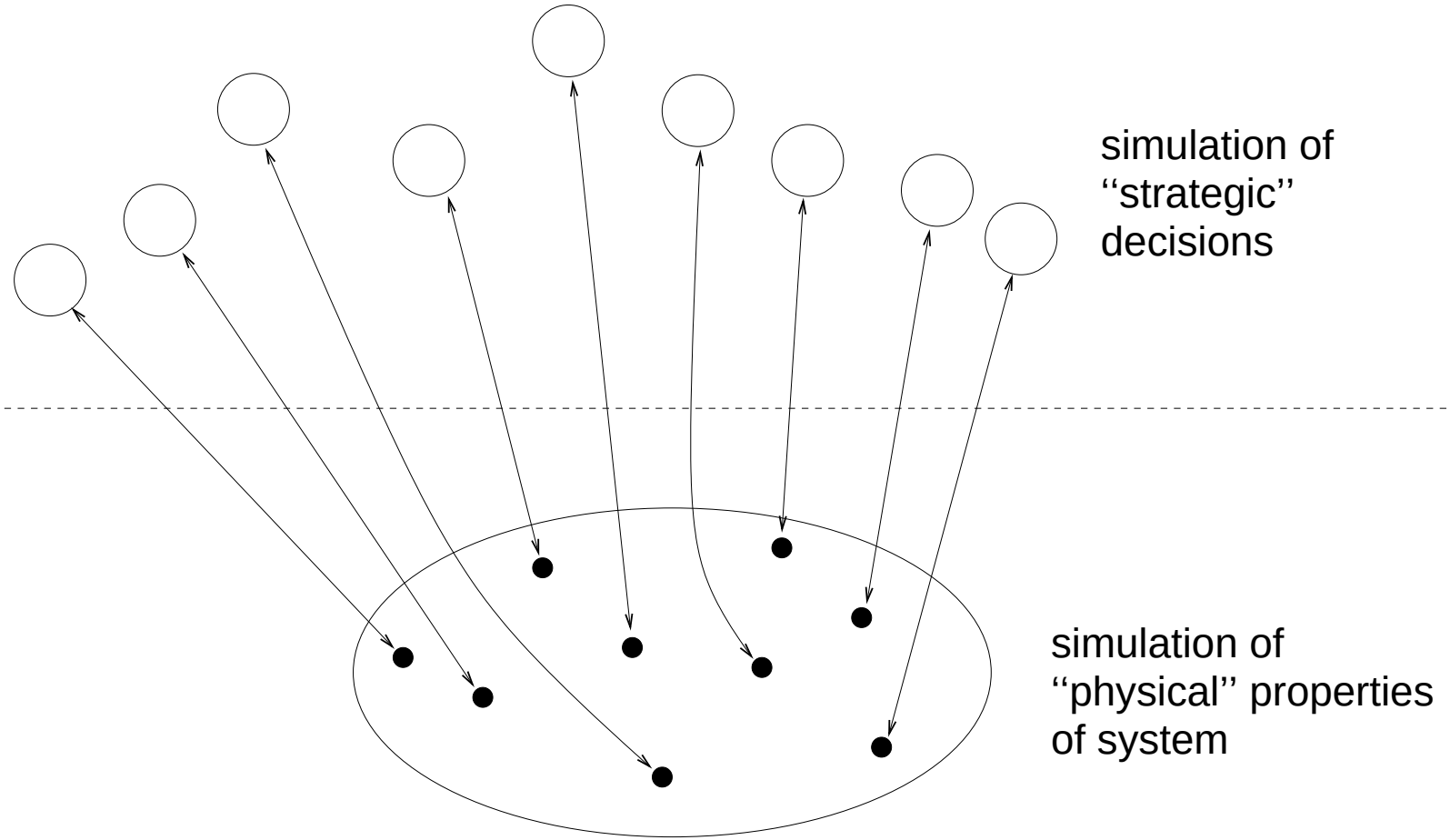
... in principle. :–)

In practice, limits because of

- coding
- knowledge
- data needs

How does a micro-simulation of travel behavior work?

Physical vs. strategical level



Different focuses:

- Strategical level: psychology, sociology, AI
- Physical level: engineering, physics

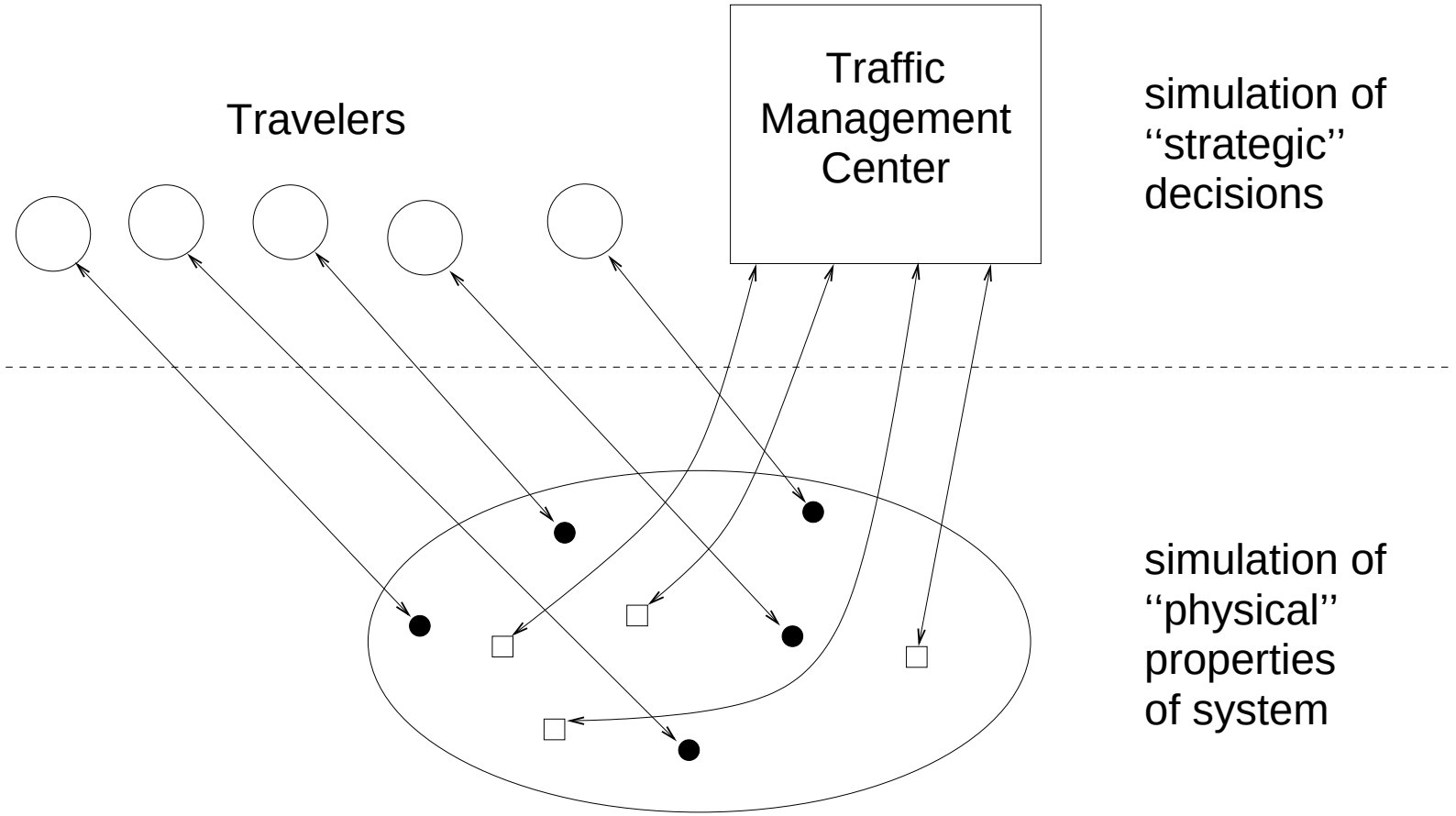
Physical vs. strategical level, ctd

Physicists tend to concentrate on the physical layer.

Computer scientists (AI) tend to concentrate on strategies.

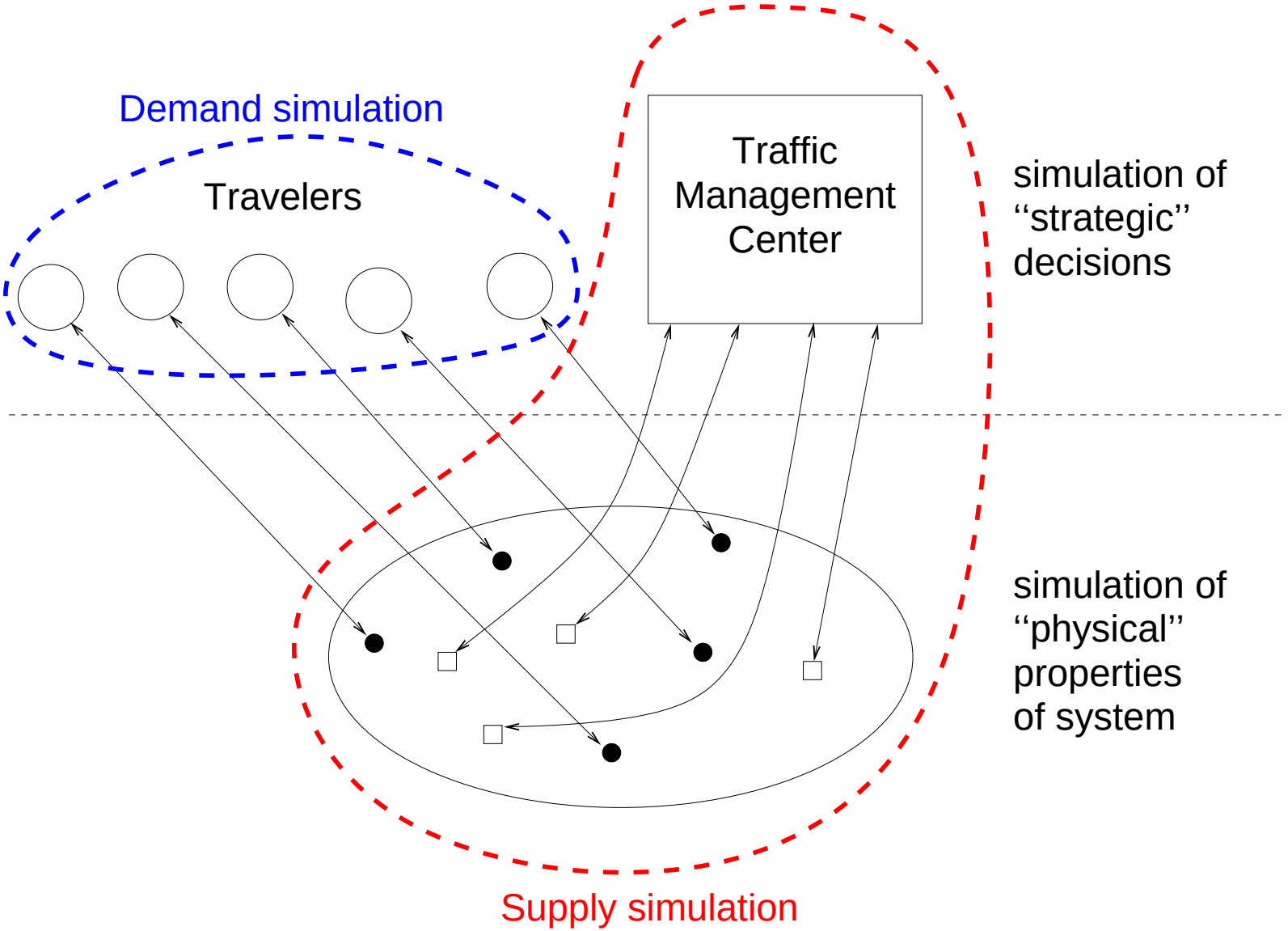
Need to combine both!

Add a traffic management center



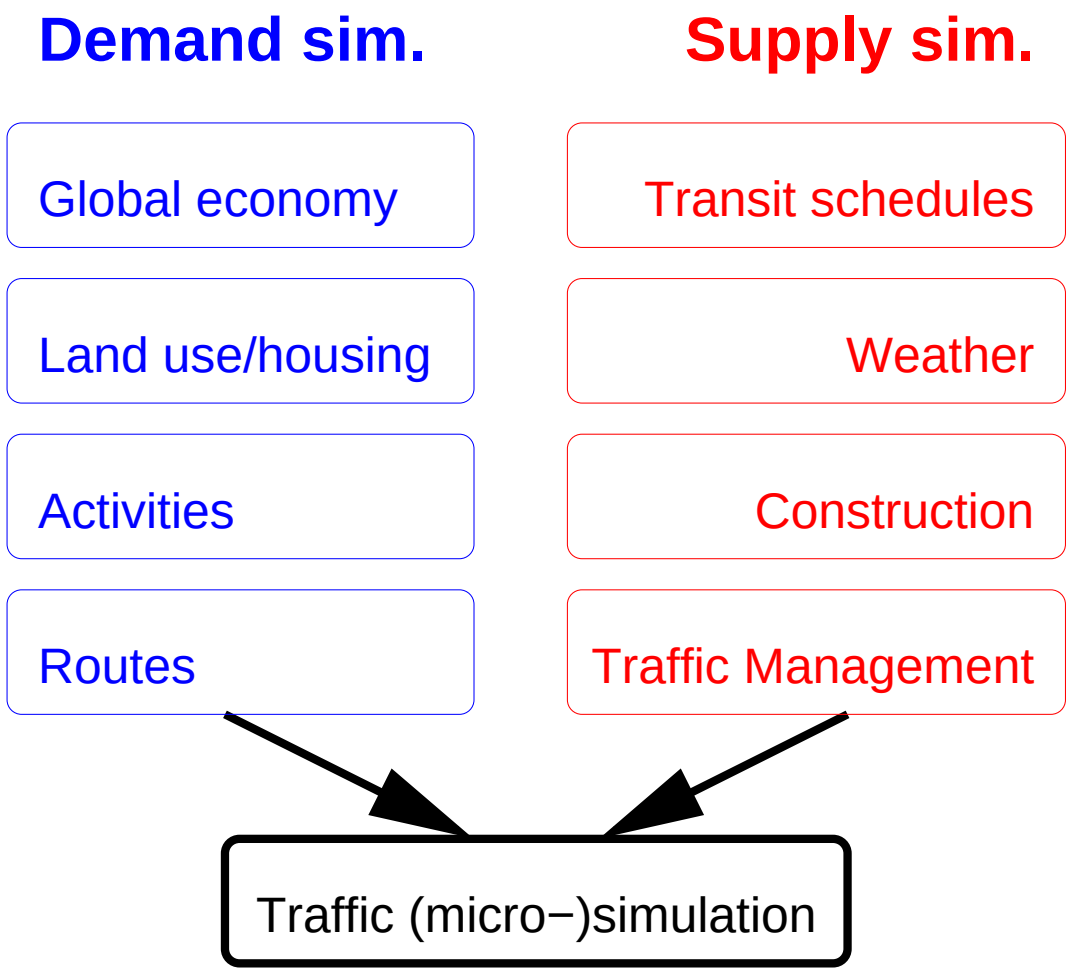
Traffic management center is just another strategy generator for objects in the simulation.

Demand/supply simulation



This does not make sense from a computational perspective.

Demand – Supply – Moderator



Physical simulation as moderator between demand and supply side – better (in my view).

Physical simulation (= mobility simulation)

Physical simulation techniques

Particle methods

- Coupled diff'l eqns. (Herman et al, Newell, Bando, ...)
- Cellular automata (Nagel, Schreckenberg, ...)
- Coupled maps (Gipps, Krauss, ...)

Field methods – partial differiential equations. (Lighthill-Whitham, Payne, ...)

Not useful since indiv particles not maintained.

Smooth particle hydrodynamics – individual particles maintained but moved according to macroscopic law.

Queuing network simulations (from OR)

Traffic micro-simulation

Can do realistic traffic micro-simulations (TRANSIMS, CA-based particle method):

Even more realistic: VISIM, PARAMICS, MITSIM, ...

Traffic micro-simulation, ctd

Sometimes, “very realistic” is too slow. Then use simulations which have less detailed dynamics:

E.g.: DYNAMIT, DYNASMART, DYNEMO, NETCELL, queue simulation. (Elements from smoothed particle hydrodynamics and from queueing network simulations.)

How much time to simulate 24 hrs of car traffic in all of CH?

2 minutes : –)

(Queueing simulation with jam spillback added; 64 Pentium CPUs with Myrinet communication; see talk Cetin)

Strategy generation

Strategy generation

Traditional: System ***is*** at state where no agent can improve (**Nash Equilibrium**).

But: This is **behaviorally not plausible**.
(Alternatively: How does the system get there?)
(Alternatively: There may be several NE.)

$\Rightarrow$ model **learning agents**

$\Rightarrow$ Two **problems**:

- (Single) agent learning
- System dynamics when all agents learn (co-evolutionary dynamics)

Single agent learning

Possible: At each “decision point”, agent computes **good/best next move(s)**.

But base future behavior on **which information**?

Trad: “System” knows answer under assumption that everybody is optimal.

We: **Iterated learning** (assume “this wednesday $\approx$ last wednesday”). Run system many times (“iterations”) and have agents learn from one iteration to next.

$\Rightarrow$ problem of Artificial Intelligence (AI)

Q Learning

Roughly:

- There are **states** (e.g. `curr_time, curr_act, curr_loc, curr_duration, acts_already_done`)
  and **agent-selectable transitions** between states (e.g. by selecting next act)
- Agent **samples** *all* possible transitions ...
  ... and eventually learns good path through state space.

Problem: Our state space ***much*** too large to learn via this.

Classifier system

Roughly:

- Several **states are grouped together** (e.g. `time = 15h-18h, act = work, duration = 7h-9h, acts_already_done = any_except_leisure`). These are the **conditions**.
- All states which fulfill the condition are mapped into the same **action** (e.g. `w/ proba 5% goto leisure`).
- Heuristic (e.g. GA) is used to **generate new condition-action pairs**.
- Condition-action-pairs are evaluated with **scores (utilities)**.

Use knowledge

Problem with classifier system: Memorizes performance via the scores for c-a-pairs. New c-a-pairs are generated randomly, or via mutation/crossover from existing ones. ***Slow.***

If we know more about the **structure** of the problem (and how humans deal with it), we can use that. E.g.:

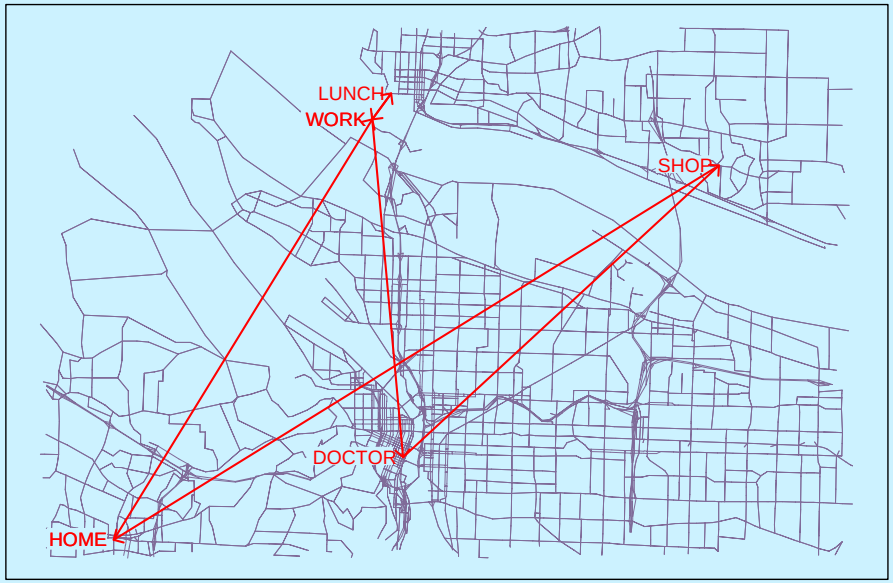
Compute fastest path on what the agent knows (mental map)

– instead of –

- Explore local moves at intersections (Q-learning)
- Random paths (classifier system)

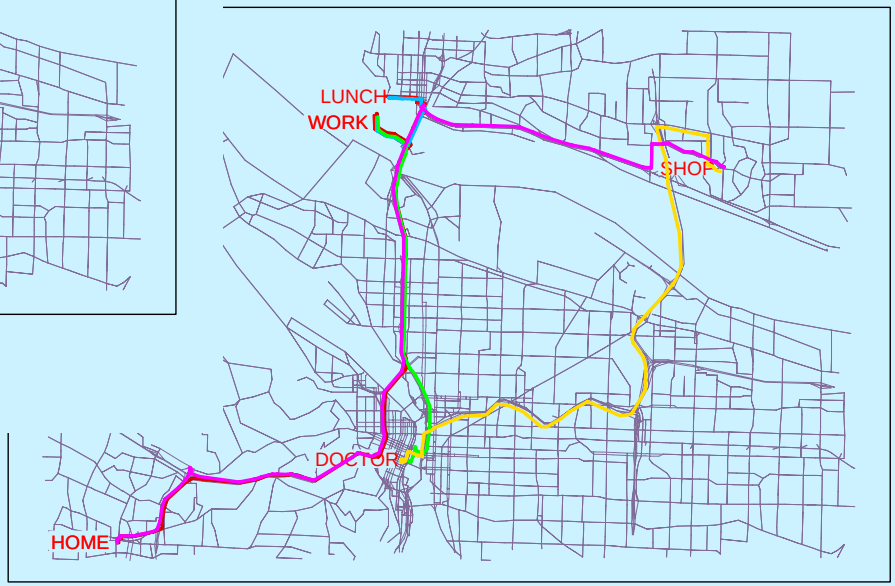
Use knowledge: Acts, routes

HUSBAND'S ACTIVITIES



<- Plans for activities

HUSBAND'S ROUTES



Plans for routes ->

Individual vs global knowledge

- E.g. give agent “fastest path based on last iteration”.
- Somewhat dishonest, because this uses **global knowledge** that real-world agent would not have (except with ITS).
- Can interpret as “rnd” new c-a-pair in classifier systems, i.e. one that the agent eventually would also find on its own. (After infinitely long learning, all these methods are the same.)

Summary single-agent learning

- Many possible approaches, relatively little structure.
- Methods which have best theoretical foundation (Q-learning) do not work in practice.
- Methods which explore structural knowledge are (so far?) somewhat orthogonal to AI methods.

Dynamics of the learning system

Deterministic learning system

All agents learn concurrently. Since this is a dynamical system, it will eventually go to an **attractor**.

For a **deterministic** system,

- attractor could be **fixed-point**, **periodic**, or **chaotic**.
  - ***Good news: A fixed-point attractor "usually" is a Nash Equilibrium (evolutionary game theory).***
  - ***Good news: In this case, can use recently developed sophisticated methods (Crittin/Bierlaire).***
  - ***Bad news: Attractor does not have to be fixed point.***
- There is a **basin of attraction** for each attractor.

Stochastic learning system

If the system is **stochastic**:

- Attractor is **stationary phase space density** (agents settle on fixed set of strat's with fixed proba's).
  This is sometimes considered good news, but see below.
- Attractors can be translated from det to stoch:
  fixed-point $\rightarrow$ "fuzzy ball" in phase space
  periodic $\rightarrow$ probability flow in phase space
  chaotic $\rightarrow$ "chaotic"
- Can jump between basins of attraction.
  - ***Bad news: This can take very long (broken ergodicity).***

Time scales

More bad news: Time-scales matter.

E.g.: If the traffic management center adapts faster than the travelers, the result is different than if it is the other way round.

(More intuitively:
Variable message signs changing from one day to the next have different effect than ...
... variable message signs which are the same at given time-of-day, and where this sequence is only slowly changed.)

Dynamics of the learning system, summary

- With luck, the system goes (close) to a Nash Equilibrium. With even more luck, this attractor is unique (cf. unique solution of static assignment).
- Without such luck, there is (currently?) very little we can say.

Dynamics of learn'g system, summary, ctd

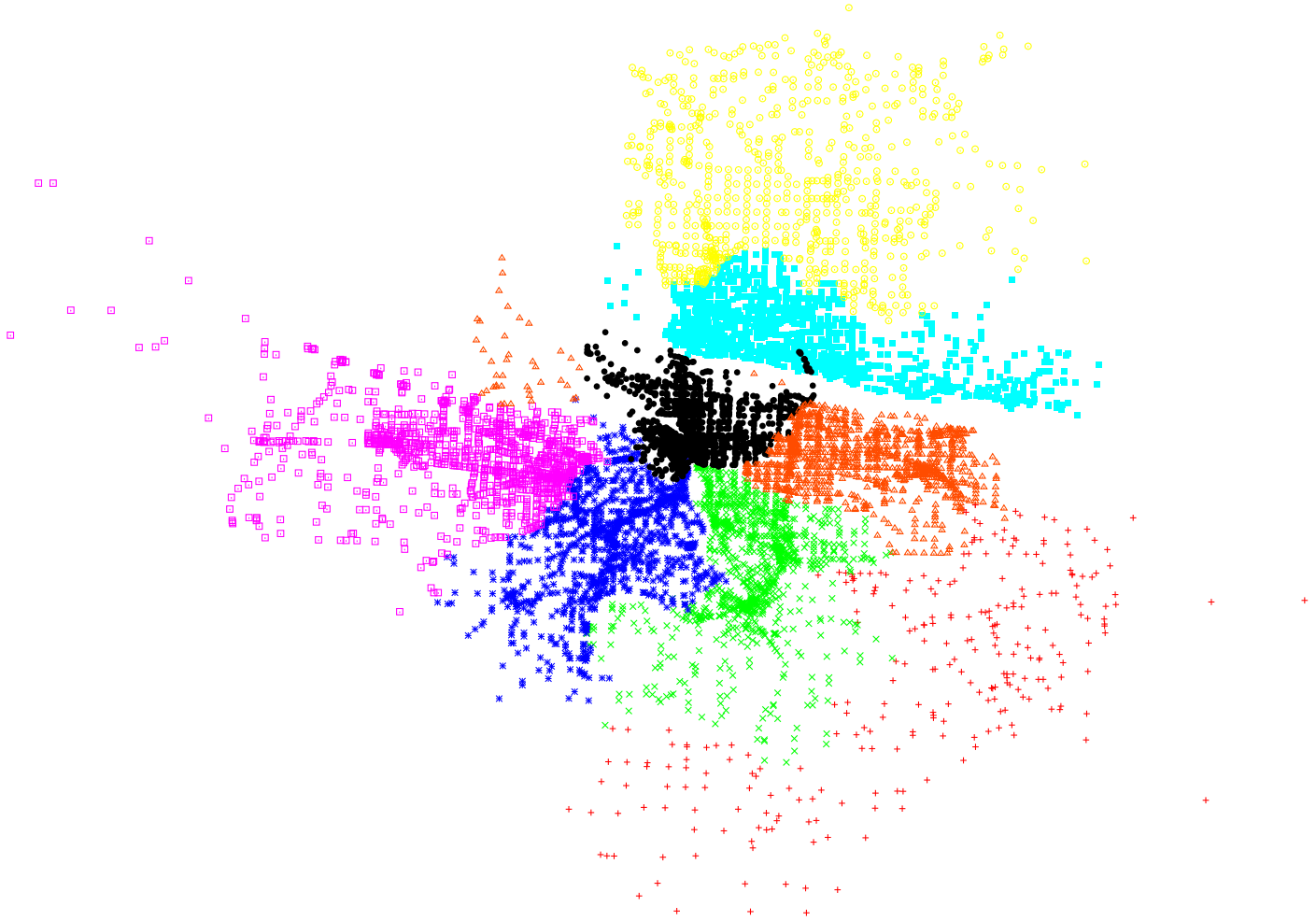
Consequences:

- Need to understand into which of the **categories** our systems fall. (E.g.: Empirically, Dynamic Traffic Assignment seems to fall into the "lucky" category. Proof??)
- If a system falls into an "out-of-luck" category, we need to understand the **stochastic basins of attraction**, how easy it is to jump out, and **how relevant this is for the real world**.
- Need to understand the issue of **adaptation speeds**, and get real-world data for this.

Computation

Computation

Have already said: Mobility/physical simulation can simulate 24h of CH in 2 min on parallel computer.



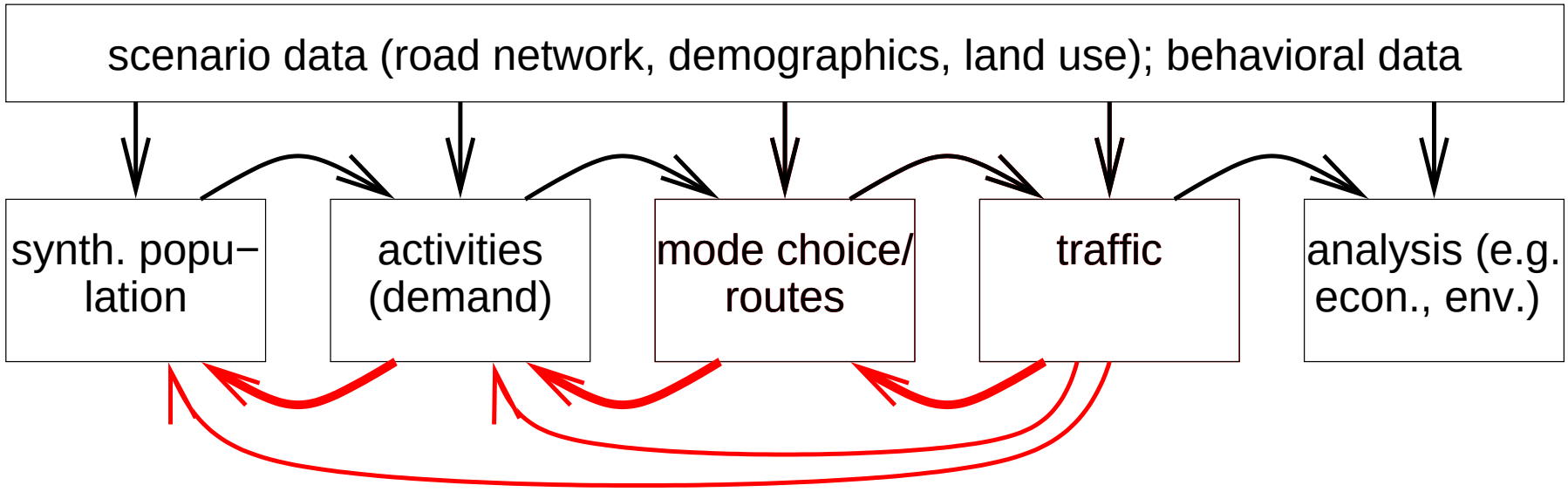
(METIS; Portland/OR)

Computation, ctd

Single-agent learning can also be distributed, but

- we are still struggling with this, and
- this will be harder once agents start to communicate (e.g. within households, ride sharing, etc.)

Third issue is **coupling between the modules**:



See next.

Simulation coupling, which information?

(cf talk Marchal, “common indicators”) My own proposal (similar to his): Two types of data:

(1) **Snapshot data**. Snapshot of vehicle positions (and maybe other things) in given time intervals.

Advantage: Visualizer output, so it is often already there.

(2) **Event data**. Data triggered by event, such as vehicle dep/ar, veh entering/leaving link, etc.

Advantages:

- Easy to implement (no aggregation at all; aggregation left to analyst).
- Nearly everything of interest can be computed from this (including, say, link travel times by time of day).

Simulation module coupling

Three options:

- based on files
- as subroutines
- based on messages

Simulation coupling via files

- Strategy generation module(s) **write strategies to file**.
- Physical simulation **reads strategies**, executes them, and **writes performance information** to file.
- Strategy generation module(s) **read performance info**, adapt strategies, and **write** them to file.
- Etc.

Advantages:

- Relatively easy to implement.
- Should work across different operating systems (win-lin).
- Corresponds to evolutionary game theory.

Simulation coupling as subroutine calls

Strategic modules are part of the simulation, e.g.:

- Move physical simulation one time step forward.
- Go through all agents and check if they want to do some strategy computation.

I do not like this very much because it will result in bloated and slow code.

But who knows ...

Simulation coupling via messages

Physical simulation and strategy simulation are in continuous exchange via messages (peer-to-peer computing).

Simulation coupling via messages

Physical simulation and strategy simulation are in continuous exchange via messages (peer-to-peer computing).

Advantages:

- Is **naturally distributed**.
- Leaves **existing modules** (e.g. route gen, act gen, ...) intact except for messages.
- Allows **“within-day replanning”**.
- Will eventually also work **across operating systems**.

(see talk Gloor, tomorrow, about “hiking”)

Computation, summary

- Simple simulation of physical system can be run **fast** enough to allow research/application even for large systems.
- Simulation architecture should support **plug-n-play** of different modules written by different groups. This is working to different degrees.

The real world

All of Switzerland

(This was: talk Raney, this morning)

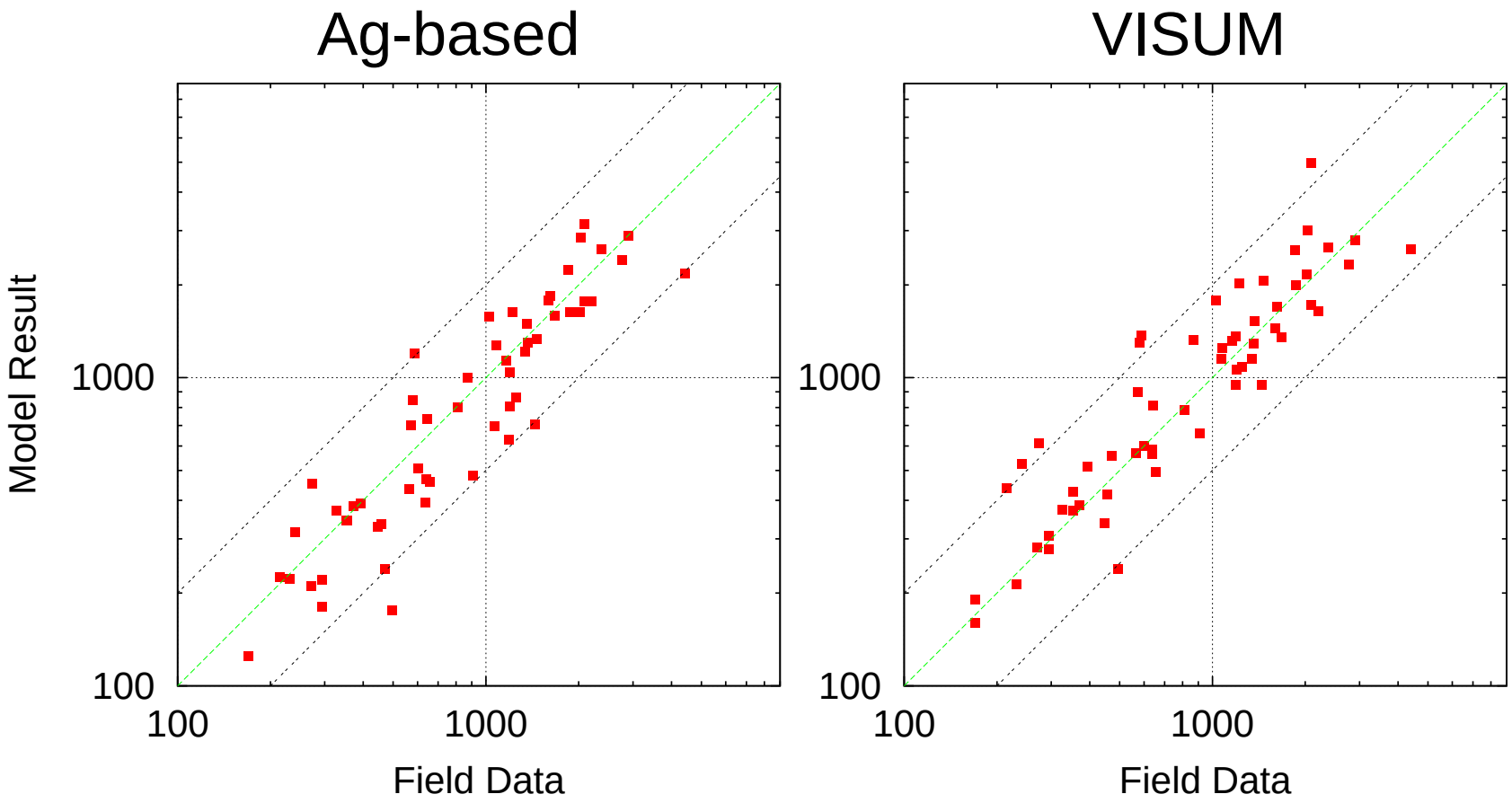
- Mobility simulation: queue.
- Strategies: routes only, time-dep fastest path, agent db.
- Demand: OD matrices.
- All of CH (6am – 9am).

Essentially a Dynamic Traffic Assignment scenario, but strictly agent-based, and large.

VISUM (assignment software) was run for comparison.

[[vis]]

Volumes 7am to 8am compared to real world



	Ag-based.	VISUM.
Mean Rel. Bias:	−5.3%	+16.3%
Mean Rel. Error:	25.4%	30.4%

Volumes 7am to 8am compared to real world, c

- Ag-based better than VISUM.
- And that is in spite of fact that OD matrix was calibrated against field counts for VISUM.

Real-world projects, near future:

- Very **high resolution network** Zurich area (w/ IVT-ETHZ).
- **Demand generation** truly agent-based (w/ IVT-ETHZ).
- Improved agent **learning** (w/ F. Marchal, CoLab-ETHZ).
- Similar approach to "**hikers in the Alps**" (w/ NSL-ETHZ, talk Gloor tomorrow).
- **Photo-realistic viewer** all-of-CH (w/ NSL-ETHZ).

Summary

Summary

- **Multi-agent simulations of large scale areas** computationally possible *and* better than assignment.
- Separation into **physical/mobility simulation** and **strategy generation**.
- Physical/mobility simulation somewhat understood.
- **Single-agent learning** ($=$ demand generation) many options, but no “standard” methods for travel behavior simulations except for route choice.
- **Dynamic behavior of the learning system** not well understood. However, for existing applications (Dynamic Traffic Assignment, DTA) probably not a problem.