Streets4MPI
(Parallel Programming Project)

Julian Fietkau
Joachim Nitschke

University of Hamburg

April 4th, 2012

Agenda

Project task, revisited

Decide on a problem that may be solved using parallel processing, and implement a solution. → **Street traffic simulation**

Main Caveat

Realistic traffic predictions can only be made using an exceedingly detailed model. **This makes things prohibitively complicated.**

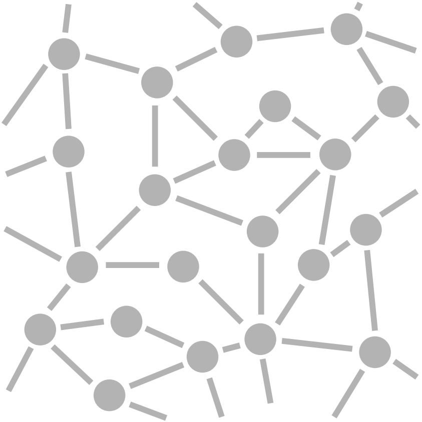
Streets4MPI is here

- We can simulate thousands of cars on the streets of Hamburg:
  - We use OpenStreetMap data for navigation.
  - We incorporate shortest-path algorithms.
  - We model road congestion and its effect on the actual driving speed.
  - We optimize the road system based on which roads are heavily used and which one are empty.
  - We can visualize all of this as dynamic heatmaps.
- Also, we can do most of this in parallel using MPI.

Revision: Discrete macroscopic simulation

- Simulation runs in steps: Traffic load in one step influences the driver's behavior in the next step
- Abstract from single cars, traffic lights etc. to daily traffic
- Display traffic development over longer time periods and influences on street network

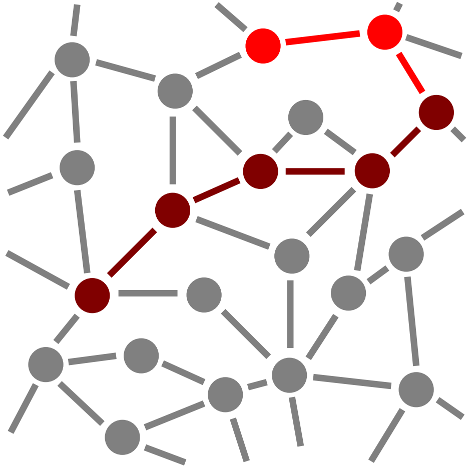
Street network



Trips

- Representation for a resident's daily traffic
- Shortest path between two nodes

Trips



Traffic load: Effect on driving speed

- Heavy traffic slows cars down
- How can we calculate the deceleration?
- Assumption: Cars keep safe braking distances

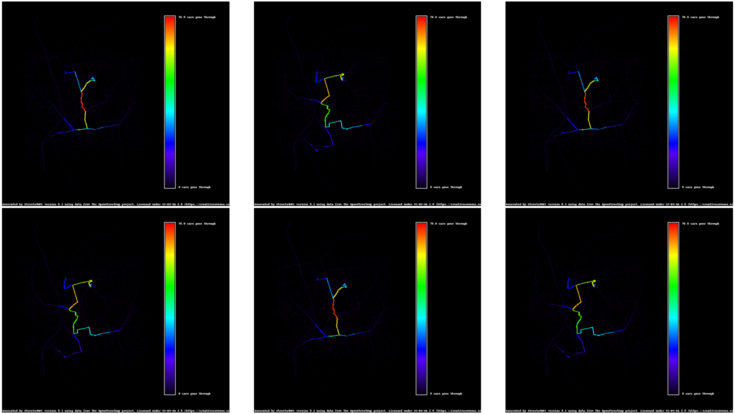
⇒ By looking at the braking distance, we calculate the actual speed

Braking distance and actual speed

$$l_{braking} = \frac{l_{street}}{n_{trips}} - l_{car}$$
$$v_{actual} = \sqrt{l_{braking} \cdot a_{braking} \cdot 2}$$

Oscillation

- Problem: Drivers show oscillating behavior



Idea

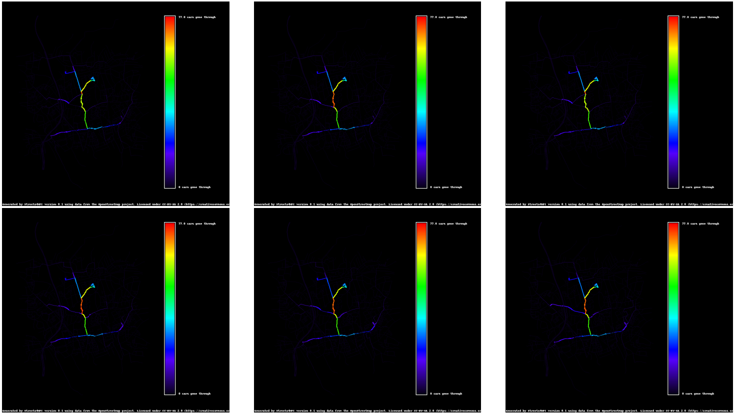
- Solution: Each driver gets an individual *traffic jam tolerance*

Traffic jam tolerance

$$v_{perceived} = v_{actual} + (v_{ideal} - v_{actual}) \cdot f_{tolerance}$$

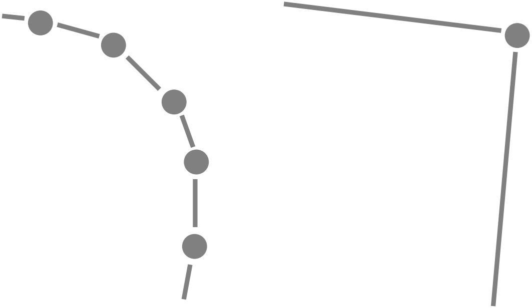
- Compromise: We assign a (random) traffic jam tolerance to each process and all its residents
  - This is because one copy of the street network graph can accomodate one traffic jam tolerance factor, and each MPI process has its own copy anyway

Result



Improvement: Reworking redundant nodes

- Problem: Redundant nodes are used to model curved streets
- Solution: Merge edges



Python

- Streets4MPI is written in Python (2.6 compatible)
- External modules: imposm.parser, pygraph, mpi4py, PIL

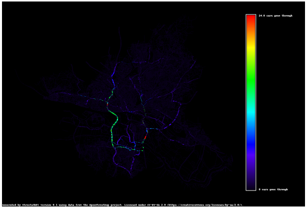
OSM parser

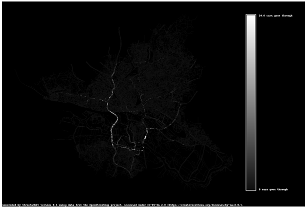
- Import data from OpenStreetMap
- XML-based semi-structured data format: OSM
- May provide additional information: street types/sizes, speed limits, residential/industrial/commercial zones

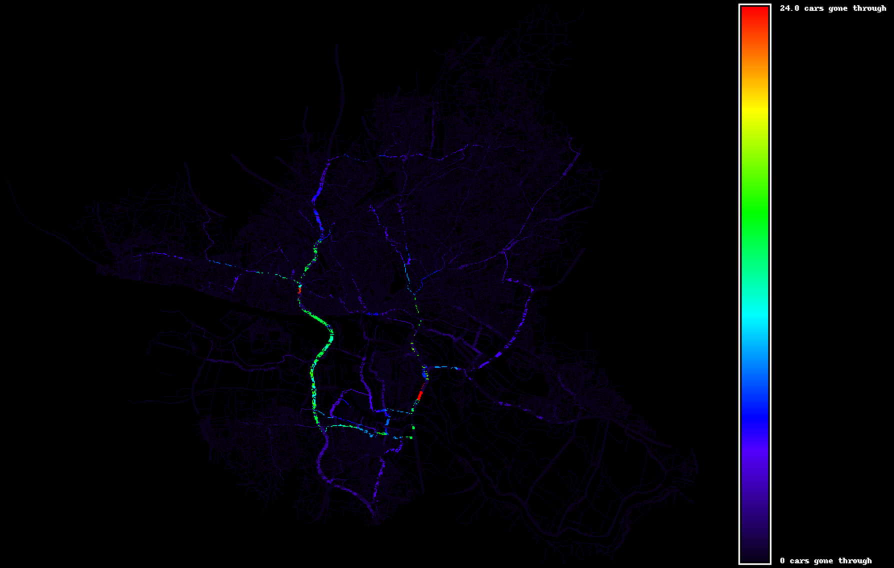
```xml
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="CGImap 0.0.2">
 <bounds minlat="54.0889580" minlon="12.2487570" maxlat="54.0913900"
    maxlon="12.2524800"/>
 <node id="298884269" lat="54.0901746" lon="12.2482632" user="SvenHRO"
    uid="46882" visible="true" version="1" changeset="676636"
    timestamp="2008-09-21T21:37:45Z"/>
 <node id="261728686" lat="54.0906309" lon="12.2441924" user="PikoWinter"
    uid="36744" visible="true" version="1" changeset="323878"
    timestamp="2008-05-03T13:39:23Z"/>
 ...
 <node id="298884272" lat="54.0901447" lon="12.2516513" user="SvenHRO"
   uid="46882" visible="true" version="1" changeset="676636"
   timestamp="2008-09-21T21:37:45Z"/>
 <way id="26659127" user="Masch" uid="55988" visible="true"
   version="5" changeset="4142606" timestamp="2010-03-16T11:47:08Z">
  <nd ref="292403538"/>
  <nd ref="298884289"/>
  ...
  <nd ref="261728686"/>
  <tag k="highway" v="unclassified"/>
  <tag k="name" v="Pastower Straße"/>
 </way>
 ...
</osm>
```

Visualization

- Simulation and visualization run independantly
- Visualization uses the **Python Imaging Library** to render the map and traffic load data to image files
- Supports many different data modes and two color modes (heatmap and grayscale)






24.0 cars gone through
0 cars gone through

mpi4py

- Object oriented interface on top of the MPI specifications
- Provides all usual MPI routines

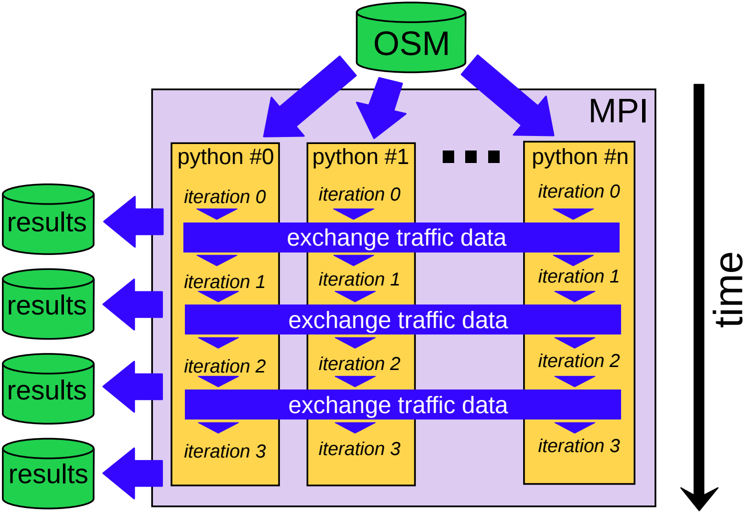
```
communicator = MPI.COMM_WORLD
object = None
if communicator.Get_rank() == 0:
   object = Object()
object = communicator.bcast(object, root=0)
```

- Single program multiple data (mostly)
- MPI code contained within our main class

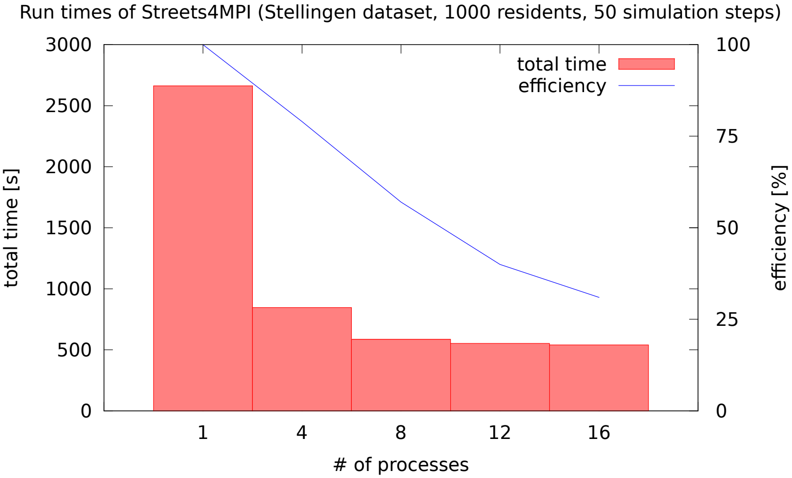
Algorithm

- Each MPI process. . .
  - . . . generates its own copy of the street network
  - . . . generates trips for its (equally divided) subset of all residents
  - . . . gets its own traffic jam resistance
  - . . . calculates the shortest paths for its residents and the resulting traffic load
- After every simulation step, each process gets sent the traffic loads from all other processes (via `mpi.allgather`)
- Complete results are saved to disk by process #0

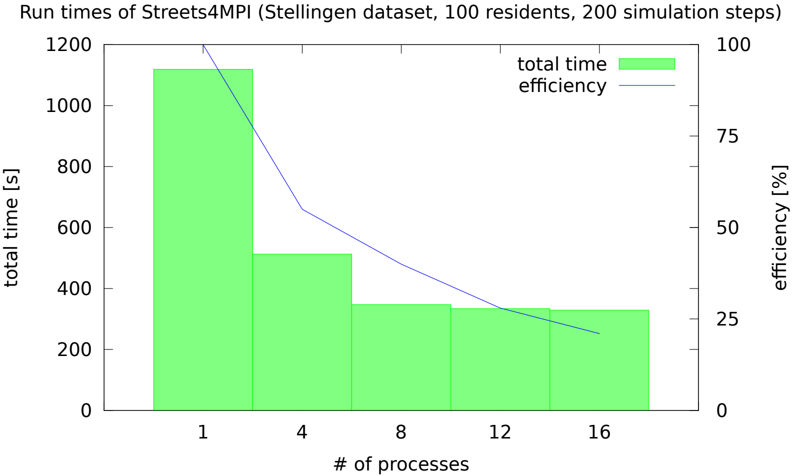
Algorithm: visualization



Performance (I)



Performance (II)



Weaknesses

- Some activities (e.g. initial I/O, road construction simulation) are not easily parallelized using our current model
- Disk activity by process #0 makes it drag behind and leave others waiting for synchronization
- Shortest path calculation is not optimal for the distributed case

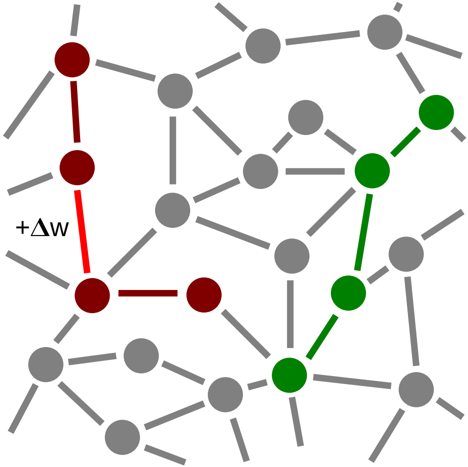
Improvement: Shortest path revisited

- Highest calculation costs are due to the shortest path calculations
- Current implementation: Dijkstra's algorithm
  - Complexity: $O(n_{nodes}^2)$
  - Executed $\sim \frac{n_{residents}}{n_{processes}}$ times
- Static shortest path vs. dynamic shortest path

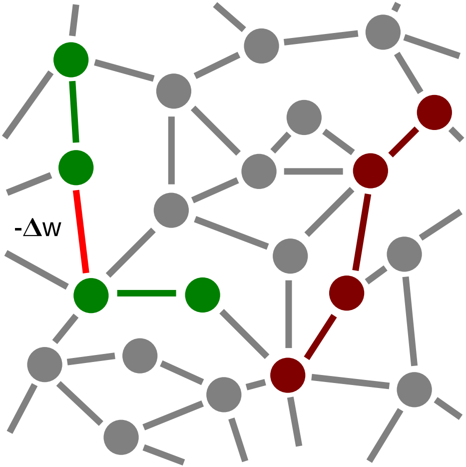
Dynamic shortest path

- Idea: Calculate shortest paths once and update them only when the edge weights change
- Performance gain through local influence of changes

Dynamic shortest path: Increasing a weight



Dynamic shortest path: Decreasing a weight



Simulation speed-up results

- Results are mostly satisfactory, but could very likely be improved
- There are constant time elements not yet parallelized
- It bears mentioning that using the current model (traffic jam resistance per process) increases simulation quality with number of processes, so real efficiency is slightly better than measured

Project Goals

- Simulation working and producing nontrivial results
- Parallel processing in Python working
- Visualization working
- Further work needed: better parallelization(?), documentation

Most Important Points

- Simple traffic simulation
- Macro level with congestion analysis, street development, visualization
- MPI on Python

Literature

WEBER, B.; MÃŒLLER, P.; WONKA, P.; GROSS, M.:
Interactive Geometric Simulation of 4D Cities
In: EUROGRAPHICS 28 (2009), Nr. 2

CHANDY, K.M.; MISRA, J.:
Distributed computation on graphs: Shortest path algorithms
In: Commun. ACM, vol. 25, no. 11, pp. 833 – 837, Nov. 1982

ANTONIO, J. K.; HUANG, G. M.; TSAI, W. K.:
A fast distributed shortest path algorithm for a class of hierarchically clustered data networks
In: IEEE Trans. Comput., vol. 41, pp. 710 – 724, June 1992

Weblinks

Project website

http://jfietkau.github.com/Streets4MPI/ (some time soon)

GitHub repository

http://github.com/jfietkau/Streets4MPI (available right now!)

Project wiki

http://pwiki.julian-fietkau.de/ (might go offline soon-ish)

Download and Usage

Download these slides and give feedback:

http://www.julian-fietkau.de/streets4mpi_final