

## SUBPART ELEVEN

# **Computational Performance Improvements**



## CHAPTER 39

# Multi-Modeling in MATSim: PSim

Pieter Fourie

### 39.1 Basic Information

**Entry point to documentation:**

<http://matsim.org/extensions> → pseudosimulation

**Invoking the module:**

<http://matsim.org/javadoc> → pseudosimulation → RunPSim class

**Selected publications:**

Fourie et al. (2013)

### 39.2 Introduction

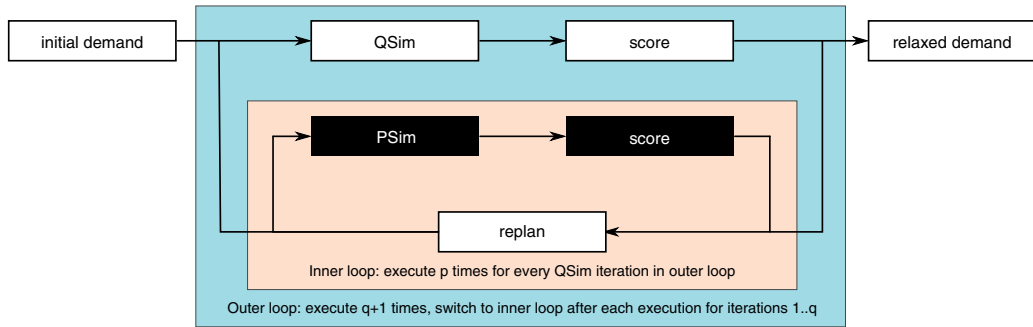
MATSim's major current performance limitation is the network loading simulation, i.e., the mobsim, for example QSim or JDEQSim; this chapter focuses on QSim. As shown earlier, QSim is repeatedly executed in the MATSim loop for the entire agent population (Section 1.2).

With the multi-modeling approach (Fourie et al., 2013), shown in Figure 39.1, a MATSim run periodically replaces QSim for a number of iterations with a simplified meta-model or PSim (Pseudo-Simulation), running approximately one hundred times faster. In risk analysis, these models are called “surrogate models” (Sudret, 2012). PSim uses travel time information from the preceding QSim iteration to estimate how well an agent day plan might perform, allowing multiple iterations of mutation and evaluation between QSim iterations to more rapidly explore the agents' solution space, producing better performing plans in a shorter time.

---

**How to cite this book chapter:**

Fourie, P. 2016. Multi-Modeling in MATSim: PSim. In: Horni, A, Nagel, K and Axhausen, K W. (eds.) *The Multi-Agent Transport Simulation MATSim*, Pp. 263–266. London: Ubiquity Press. DOI: <http://dx.doi.org/10.5334/baw.39>. License: CC-BY 4.0



**Figure 39.1:** Operation of a MATSim run implementing pseudo-simulation.

Source: Fourie et al. (2013), Figure 1, p. 69

### 39.3 Basic Idea

PSim exploits classes that record various network performance aspects during queue simulations and uses them as an approximate meta-model of QSim. It fires the same sequence of events for car and public transport passengers that is produced during a QSim mobility simulation, except that event timings are approximate values expected at the time of day they occur.

For private vehicle traffic, it calls the

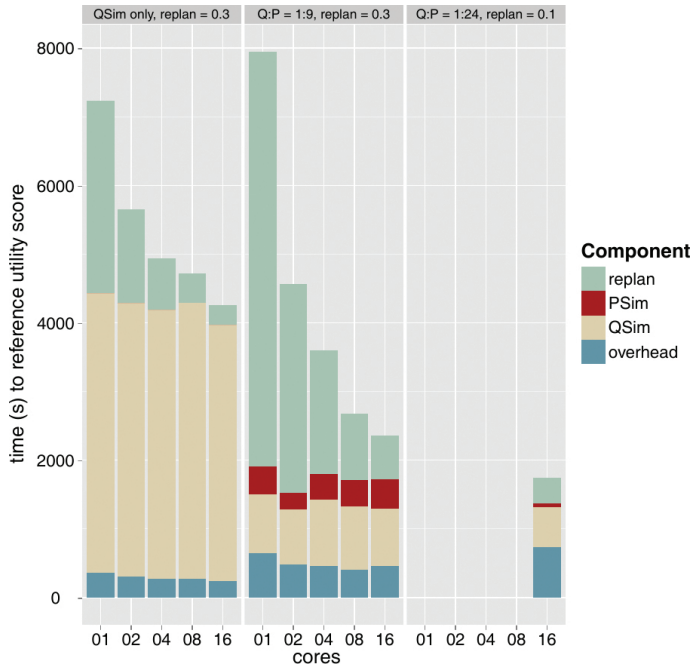
```
getLinkTravelTime(Link link, double time, Person person, Vehicle vehicle)
```

method of classes implementing the `TravelTime` interface to fire `LinkEnterEvents` and `LinkLeaveEvents` at appropriate times for all car route links. For public transportation, the events sequence generated for a passenger traveling on a particular service relies on a meta-model of stop-to-stop travel times (interface `StopStopTime`) and waiting times at stops (interface `WaitTime`); both concepts were developed by Sergio Ordóñez at the Future Cities Laboratory (package `playground.singapore.transitRouterEventsBased`).

PSim plans are scored using the same function as QSim iterations and are compatible with most replanning modules in MATSim. Following a series of PSim iterations, a plan is selected for each agent, in the usual fashion, and a QSim iteration is run to start a new cycle. The various classes used in PSim are updated with the latest network performance information and the process repeats.

### 39.4 Performance

Initial tests on the Zürich scenario (described in Chapter 56) have shown a dramatic decrease in computation times, compared to the default QSim-only approach; performance improves linearly with an increasing number of computational cores. Figure 39.2 compares the PSim-approach, in two configurations, against the existing approach, for a 10 % sample of private vehicle traffic in Zürich. All simulations were run until they reached a target score, i.e., the score reached after running the standard approach for 100 iterations. The first PSim-implementing configuration uses the same rate of plan mutation as the QSim-only approach, where 30 % of agents are selected for plan mutation (replanning) after each iteration, whether it is a QSim or PSim iteration. The new approach requires fewer QSim iterations to reach a target score, but requires more time for replanning. Replanning is fully multi-threaded, with no synchronization between cores required, so its performance increases linearly, with increasing number of cores; times improve more dramatically with the new approach than the standard approach. In the second configuration, the mutation rate



**Figure 39.2:** Computation time contributions vs. number of computational cores for QSim-only (0.3 replanning rate), 9 PSim iterations per QSim iteration at 0.3 replanning rate, and 24 PSim iterations per QSim iteration at 0.1 replanning rate.

Source: Fourie et al. (2013), Figure 4, p. 73

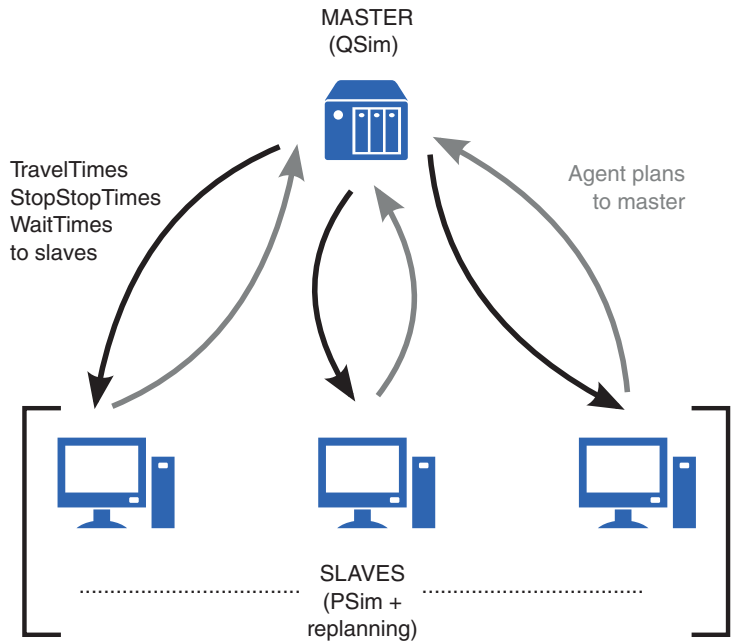
is reduced and the number of PSim iterations between QSim iterations increased to 24 for each QSim iteration. The system now tests many more combinations of different mutation operations (four in this case: activity timing, mode choice, secondary activity location choice, and re-routing), to reach the target state much faster, even though it produces a smaller expected number of mutated plans per agent between QSim iterations (three for configuration 1, 2.5 plans for configuration 2).

This last point raises an interesting issue; namely, that the distribution of mutation operation numbers can be dramatically spread out with the PSim approach, because increasing the number of iterations is relatively cheap. This should make the approach preferable, especially with random mutation-producing replanning strategies, where a large number of mutations are needed to produce a relaxed simulation state.

For a detailed discussion of the meta-modeling approach and the results of applying this method to the Zürich scenario, refer to Fourie et al. (2013).

### 39.4.1 Distributed Computing

Because PSim executes plans independently from each other, requiring no coordination of computational processes, it is possible to distribute it across multiple nodes, with no need of shared memory, as illustrated in Figure 39.3. To this end, we (Fourie and Ordóñez, FCL (Future Cities Laboratory)) are implementing a simple messaging protocol to transmit network performance objects to PSim slave nodes from a master node running QSim only. Slave nodes perform replanning operations and evaluate plans in a pre-determined number of PSim iterations per cycle. At the start of each QSim iteration, a single plan for each agent is transmitted back to the master from



**Figure 39.3:** Master-slave configuration for running PSim in distributed mode, across many slave computer nodes in a local area network or in a cloud computational framework. The master runs selected plans in a full queue simulation and transmits updated travel time information to slave nodes after every iteration. In turn, slaves produce and evaluate new plans in repeated PSim/replanning cycles, sending the master a single plan for each agent at the start of a QSim iteration.

all the slaves, and updated `TravelTimes`, `StopStopTimes` and `WaitTimes` are rendered during the full mobility simulation, to be transmitted back to the slaves in the next cycle.

The approach yielded promising results, with a reduction in the number of QSim iterations, as in the previous work, as well as the potential for running large-scale simulations on much cheaper hardware than the current approach, that demands expensive shared memory servers. Most importantly, all replanning takes place in parallel with the QSim running on the master, so the time spent waiting for replanning operations can be reduced to nil. This performance increase is especially useful for large scenarios implementing public transportation, where the time spent replanning can be up to twice that of the queue simulation.