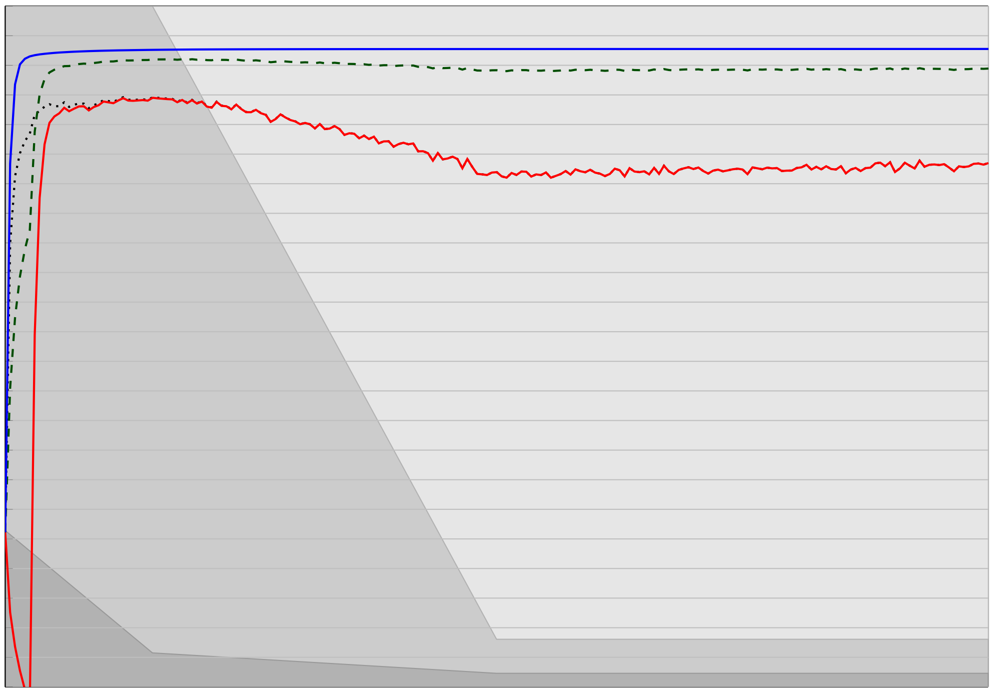


Optimization of the convergence of an agent-based simulation

Sylvain Wenger

Term Project

January 2011

Institut für Verkehrsplanung und Transportsysteme
Institute for Transport Planning and Systems

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Contents

List of Figures

List of Tables

Term Project

Optimization of the convergence of an agent-based simulation

Sylvain Wenger
IVT
ETH Hönggerberg
CH-8093 Zürich
sywenger@student.ethz.ch

January 2011

Abstract

The aim of this term project is to develop new strategies for the agent-based simulation tool MATSim in order to increase the convergence rate and also the maximum score at the equilibrium. A new module uses the ReRouting, TimeMutator and SelectExpBeta strategies with probabilities that depend on the current iteration number; the importance of each module and in particular of the SelectExpBeta module are also discussed.

Keywords

MATSim,convergence,agent-based simulation,iteration-dependent probabilities

Preferred citation style

Wenger S. (2011) Optimization of the convergence of an agent-based simulation, *Term Project*, Institute for Transport Planning and Systems (IVT), ETH Zurich, Zurich.

Arbeitsberichte Raum- und Verkehrsplanung

Optimization of the convergence of an agent-based simulation

Sylvain Wenger
IVT
ETH Hönggerberg
CH-8093 Zürich
sywenger@student.ethz.ch

January 2011

Zusammenfassung

Das Ziel dieses Projektes ist es neue Strategien für die agentenbasierte Simulation Tool MATSim zu entwickeln, um die Geschwindigkeit der Konvergenz und auch den maximale Score im Equilibrium zu erhöhen. Ein neues Modul benutzt die ReRouting, TimeMutator und SelectExpBeta Strategien mit verschiedenen Wahrscheinlichkeiten, die von der aktuellen Iterationsnummer abhängen. Zudem wird die Wichtigkeit der einzelnen Module und insbesondere des SelectExpBeta Moduls diskutiert.

Schlagworte

MATSim,Konvergenz,agenten-basierte Simulation,Iteration-abhängig Probabilitäten

Bevorzugter Zitierstil

Wenger S. (2011) Optimization of the convergence of an agent-based simulation, *Arbeitsberichte Raum- und Verkehrsplanung*, Institut für Verkehrsplanung und Transportsysteme (IVT), ETH Zürich, Zürich.

1. Introduction

Traffic analysis and forecasting is crucial in the development and improvement of the transport network of cities and countries. Nowadays this is often done using computer simulations of traffic flow. There are many techniques that aim to describe the traffic with some formal processes or equations for example probabilistic simulation using statistical data, partial differential equations similar to fluid dynamics, and one of currently most used technique microscopic simulation that are based on the interactions between the individuals. The chair of Transport Systems Planning and Transport Telematics at the Institute for Land and Sea Transport Systems of the Technische Universität Berlin, led by Prof. Dr. Kai Nagel, and the chair of Transport Planning at the Institute for Transport Planning and Systems (IVT) of the Swiss Federal Institute of Technology Zurich, led by Prof. Dr. Kay W. Axhausen, developed a toolbox which simulates large-scale agent-based transportation problems and is constructed in a modular way such that is it easily adaptable to many scenarios. This toolbox, named MATSim, is an open-source project in active development and its modular structure permits the implementation of new modules for improving the simulation, or modelling an agent-behaviour with a different algorithm.

Normally MATSim can run on any modern laptop for small scenarios, but for large scenarios the simulation outputs a huge amount of data and requires more powerful computers. Some well-investigated scenarios concern the cities of Zürich and Berlin and due to the size of these cities, the computations need several hours or days. MATSim's core is a randomized algorithm that converges to an equilibrium between all agents such that each agent ends with a good time schedule and a routing plan for the day. The time needed for the computations depends largely on the convergence speed of this algorithm, the optimization of the convergence rate is therefore an important task in the development of MATSim. This project will focus on the mathematical development of a new strategy module which improves the convergence and not on a better implementation of the algorithms already used.

The new modules will be based on the idea that the strategies for each agent don't have to be fixed with some probabilities: the module relaxed the fact that the probabilities are constant during the whole iteration process, the probabilities will depend linearly of the number of iterations with some parameters that will need to be determined. The optimization problem will lead to an analysis of the importance of the SelectExpBeta module for the evolutionary algorithm.

2. MATSim

The toolbox MATSim was designed in order to offer a very flexible tool for the simulation of agent-based systems. Is is currently used mainly on the traffic network of Zurich and its surrounding environment, also on the traffic network of Berlin and Brandenburg state, the city of Toronto (Canada), the province of Gauteng (South Africa) and even in an emergency scenario of the evacuation of Padang (Indonesia) caused by a tsunami. Zurich was the most interesting scenario from the start since precise statistical data for the network and the population were already available, and since MATSim is developed jointly by the Swiss Federal Institute of Technology of Zurich and the Technische Universität Berlin.

This term projects aims to optimize the convergence speed of MATSim's evolutionary algorithm by implementing a new strategy module. The scenario of Zurich is far too complex to run many tests, therefore it was more simple to begin with an artificial scenario (Equinet) and later with the Berlin scenario. This scenario will be described in the section 2.1.1.

Generally, a MATSim simulation will need some input files (network, plans for the agents, objective function for the agents) which will be used to simulate the behaviour of agents on a network. This behaviour is represented as a list of events for each agents consisting of time departure and arrival, routes that the agents will use, mode choice, and other data that depend on the scenario. The aim of the simulation is to output plans and schedules for each agents such that no agents would like to change his mind and find another schedule or route. This equilibrium is nearly related to the Nash Equilibrium: in a Nash Equilibrium, each agent cannot improve his objective function by changing his choices independently of the other agents. In MATSim each agents is supposed to have a perfect knowledge of the other agents plans and of all details of the network. This assumption is highly non-realistic and can be changed by creating a mental map of the knowledge that each agent has, as described in [Bal07].

MATSim is written in Java and is distributed under the Gnu Public License, therefore it is capable of running on the main existing operating systems (Windows, Linux, Mac OS). Besides this convenient portability, it is also open source and allows to implement new modules to run more specific scenarios or to improve the performance of MATSim.

2.1 Input

The input files are written in xml format and permits a modularity in the tools used: it is easy to define new functionalities or to change the existing parameters without modifying any code of MATSim. Usually the input files consists of a network description file, a list of plans of activities for each agents with an initial schedule for the day, and a configuration file which gathers all links to the file and all important values for the simulation.

2.1.1 Network

A description of the Berlin scenario is presented in [Bal07], we will only present the main ideas of the encoding of a network for MATSim. The network file contains two different important entities: there are nodes and links. The network is hence a special directed graph with many attributes for each edge: each edge correspond to a link between two nodes, it has a length, a free speed limitation, a flow capacity (how many vehicles can pass through a link in a defined interval), and some additional parameters for the number of lanes and if it is a oneway road or not. The scenario of Berlin was preferred over the simple Equinet scenario, but due to its size the scenario was reduced to one percent of its population and to keep a realistic network, the flow capacities were also reduced to two percent. The "one-percent" reduction of the population is set by deleting agents in the initial plans file. For all links the flow capacity factor and storage capacity factor for such scaling are defined in the configuration file by

```
<param name = "flowCapacityFactor" value="0.02" />
<param name = "storageCapacityFactor" value="0.05" />
```

It is easy to transform this scenario into a congested case by decreasing both values. For an extreme case of congestion, the parameters were divided by ten.

2.1.2 Plans

The plans consists of a list of agents with their corresponding activities for a day. These plans are generated randomly prior to the use of the simulation with MATSim since a synthetic population can be generated once and used many times. The data comes usually from national institute of statistics or from more local censuses. A typical description of an individual is as follows:

```
<person id="66128" sex="m" age="20" license="yes" car_avail="always" employed="yes">
  <plan age="0" selected="yes">
    <act type="home" link="1921" x="459.71" y="582.10" start_time="00:00"
                                dur="08:00" end_time="08:00" ref_id="6" />
    <leg num="0" mode="car" dep_time="08:00" trav_time="00:33" arr_time="08:33">
      <route trav_time="00:33">
        1880 1881 1884 9322 1887 1886 1968 1966 9333 630053 630153
      </route>
    </leg>
    <act type="work" link="13816" x="456.93" y="58.11" start_time="08:00"
                                dur="08:00" end_time="24:00" ref_id="917" />
    <leg num="1" mode="car" dep_time="16:33" trav_time="00:33" arr_time="17:07">
      <route trav_time="00:33">
        630053 630054 630055 1881 1880 1879
      </route>
    </leg>
    <act type="home" link="1921" x="459.71" y="582.10" start_time="11:33"
                                dur="08:00" end_time="24:00" ref_id="6" />
  </plan>
</person>
```

As one can see, an agent is described by some personal attributes such as the gender, the age, the employment status and if this person has the possibility to use a car (licence and car available). Moreover one plan for the day is assigned to this person, each activity is defined as "home", "work", "school", "leisure" or "shopping" at a certain location and with a given timespan corresponding to the opening and closing hours of each facilities. Between each activity there is a travel defined by a mode (car, walk, public transports...), a departure and arrival time and an initial route.

The routes and the time schedule of each agent will be modified throughout the iterations of MATSim's evolutionary algorithm.

2.1.3 Configuration file

- Utility function: Each agent is now basically described, but how does these agents choose between their plans? How to quantify each plans with a score that represent truthfully the objectives of an agent? The choice and scores in MATSim are based on Discrete Choice Theory, that is there is a set of alternatives $1, \ldots, n$ for each agent and they have to choose the best alternative. Each alternative i has a corresponding utility function U_i, and an agent select alternative i if for all j: $U_i \geq U_j$. The utility function is usually defined by a linear combination of parameters and a random term: $U_i = \beta T_i + \varepsilon_i$, where ε_i can be chosen as a normal distribution (Probit model), an extreme value distribution (Logit model) or other distributions. The variables β are estimated by census data of a given population, and the used values are defined as follows:

  ```
  <param name="lateArrival" value="-18" />
  <param name="earlyDeparture" value="-0" />
  <param name="performing" value="+6" />
  <param name="traveling" value="-6" />
  <param name="waiting" value="-0" />
  ```

  Therefore an early departure does not cost anything for the agent but prevents him from increasing his utility by the performing of an activity.

- Memory: each agent keeps a pool of plans of a certain size so that there is a smaller risk to lose a "good" plan by creating a new one (and forget the previous one). The memory has a fixed size N and each agent can either create a new plan or select an existing plan at each iteration. If an agent creates a new plan, there will be $N + 1$ plans, hence the worst plan will be deleted. A value $N = 5$ is usually taken as a good compromise between a large enough memory for the agent and a reasonable consumption of disk storage:

```
<param name="maxAgentPlanMemorySize" value="5" />
```

- Learning: at each iteration, the selected plan of an agent will be evaluated and obtain a score S_{new}. The effective score will be calculated by $S = \alpha S_{new} + (1 - \alpha) S_{old}$ where α is a learning parameter. A value of 0 would mean that the score will remain unchanged during the process, a value of 1.0 would mean that the agent does not care about the previous score. For this project the following value will be used: `<param name="learningRate" value="1.0" />`.

- Module: the strategies that will be used in the iteration process can be initialized in the configuration file. This is very useful since one needs only to change the config.xml file in order to change the parameter that will be used:

```
<module name="changeStrategyProbabilitiesOverTime">
  <param name="routing0" value="0.23" />
  <param name="routing1" value="0.05" />
  <param name="routing2" value="0.05" />

  <param name="timeMutator0" value="0.77" />
  <param name="timeMutator1" value="0.95" />
  <param name="timeMutator2" value="0.95" />

  <param name="firstStrategyChangeAtIteration" value="30" />
  <param name="secondStrategyChangeAtIteration" value="100" />
</module>
```

  The above depicted parameters are for the newly implemented module that will be described in the next chapter.

2.1.4 Strategy modules

TimeMutator

This strategy module is very simple and just shift the activities end time randomly by some value taken uniformly in the interval $[-30, +30]$ minutes. Because of the randomness of this procedure there is no certainty that this module will increase the score of an individual, it could even drive the score from its equilibrium to a chaotic state. The main interest in this module is its simplicity which implies that it is very fast to compute and does not increase much of the time needed to perform one iteration of MATSim-EA.

ReRouting

There are several modules implemented that modifies the routes taken by some agents based on different techniques to compute the routes: Dijkstra algorithm and A^*-algorithm. These modules are very important since an agent will not have the same travel time on a empty network and on an congested network, therefore after this agents knows also the route of the other agents he can decrease his travel time by choosing another route which is less congested. Obviously if a majority of agents performs a re-routing there will be oscillations in the search of good routes since the congested area will vary much. Hence one can expect that a too large fraction of re-routing is useless in order to decrease the number of iterations. There exists several ways to compute a shortest route on a weighted graph, but all these algorithms takes much more time than the previous module since the number of calculation required will strongly depend on the size of the network.

SelectExpBeta

This module chooses randomly a plan among the already existing plans with a probability depending on the score of each plan, that is the higher the score the higher the probability to be selected.

BestSelect

Similarly to SelectExpBeta this module selects a plan in the already existing plans of an agent, but contrary to the previous module it will always select the plan with the best score.

Planomat

Due to the inefficiency of the TimeMutator module caused by its randomness, a new module has been implemented: Planomat employs a Genetic Algorithm that computes better time schedule for an agent and runs also some rerouting procedure. This module decreases the number of iteration of MATSim-E, but does not improve much of the computation time needed since one needs to perform a genetic algorithm-based optimization for each agents.

2.2 Ouput

The output is composed of general files about the run of MATSim and some files for each iteration of the simulation. There is a log file that saves all the tasks performed and all the main results, there are some data files that save the average, worst, best and executed score and that save the average, best and worst travel distances for each agent. These values can also be seen directly on a graph in the pictures of the output. At each iteration, all the plans with the chosen routes and schedule are saved so that the simulation of the behaviour of the agents can be analysed and observed at each iteration.

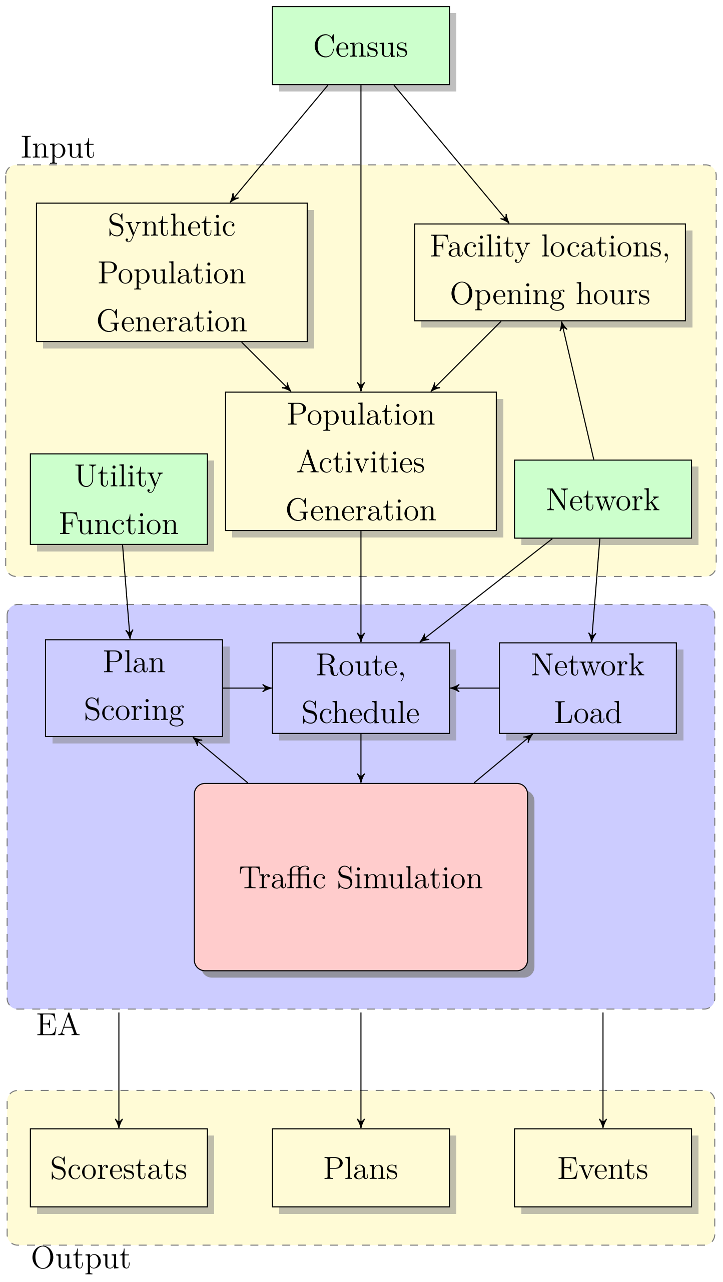
2.3 Procedure

The core of MATSim is the MATSim-Evolutionary Algorithm (MATSim-EA) which is similar to a genetic algorithms by its iterative procedure and its randomness, but with some additional features and not exactly the same iterative way to improve the objective function. In MATSim, each agent has its own plan and will not share them with other agents but at each iteration the environment for each agent is updated according to the choices of the agents in the previous step and the agent will create, or select among his pool of plans, a new plan in order to improve his score.

First of all, the agents defined by the input files have all a schedule for their activities of the day and an initial guess for the plan (which represent how each agent will accomplish his travels). These plan will be evaluated by the simulation in MATSim-EA. The simulation is based on a stochastic queue-based approach: each link works as a queue, the first coming in is the first going out. Each link has a fixed storage capacity based on its length, so a queue can create congestion if the traffic is to high on a single link. After the simulation, each agent's plan gets scored according to the utility function defined in the configuration file. At the next iteration, agents will select randomly a plan in their memory or create a new plan using one of the strategy module. The network load is updated with the data of the last iteration, and therefore the re-routing will avoid the congested areas and improve the time needed for the travels. If an agent has more plans than the allowed size of the memory, the worst plan (in terms of score) is deleted. At the end of each iteration, the events (routes taken and timetables) are saved in an output file.

The general procedure can be seen in the following graph. Each arrow represent the order of execution. For more details on the structure of MATsim see [Bal07] and [Mei10].

Figure 1: Diagram of the structure of MATSim



3. Iteration-dependent probabilities

One of the first idea for a new strategy module is to remove the constraint that the probabilities for the strategy choice are fixed during the whole process: it could be useful to have large probabilities at the beginning since we are far from the equilibrium and a lot of agents have to change their strategies, we also would like to have small probabilities at the end because we want to obtain a stationary state (equilibrium) which cannot be true if we ask too many of the agents to change their plans.

The aim is to simulate the Berlin scenario with the same strategies modules as in [Apt10] but with some probabilities that will depend of the iteration number.

3.1 Configuration

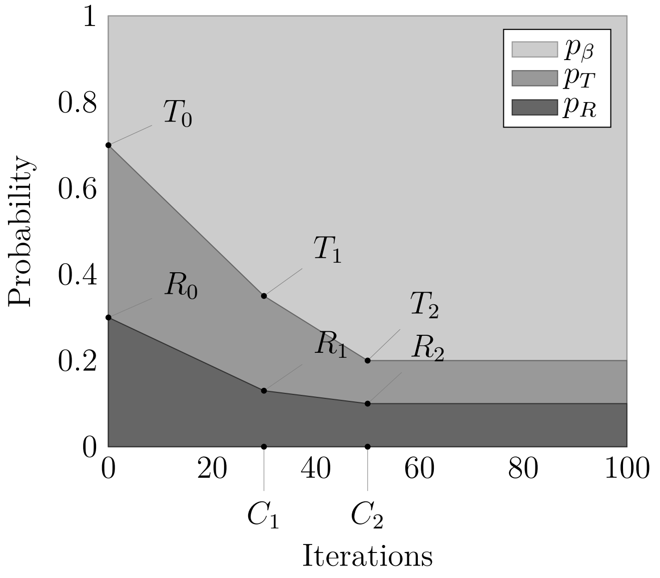
At the beginning of the work, it was planned to simulate the Equinet scenario which is a very basic network. The principal interests in this scenario is its small size which implies short simulations and the symmetric construction of the network which makes this scenario very precisely controllable.

After some test on the cluster of computers of the ETHZ, Brutus, the simulation of a more realistic scenario was selected: for all tests the Berlin scenario reduced to 1% of its population and 2% of its flow capacity is used (except for the analysis of congested scenario where the flow capacity is decreased).

3.1.1 New module

The module `changeStrategyProbabilitiesOverTime` is the main focus in the next chapter and corresponds to a distribution of probabilities for a selection of strategy modules. Each module is selected randomly according to a linear spline function. The module has 8 parameters: each probability fraction is represented as a spline of 3 lines, the first line is drawn between 0 and C_1, the second line between C_1 and C_2, the third one between C_2 and the last iteration number. The spline is defined by its values at iteration number 0, C_1, C_2 with respectively the numbers T_0, T_1, T_2 for the TimeMutator module, or R_0, R_1, R_2 for the ReRouting module. There is no need to specify the probabilities for the SelectExpBeta module since the sum of the probabilities should always be one.

Figure 2: Parameters for the new module



For example the probability p_T of the TimeMutator strategy is given by

$$p_T(n) = \begin{cases} T_0 \cdot \left(1 - \frac{n}{C_1}\right) + T_1 \cdot \frac{n}{C_1} & n < C_1, \\ T_1 \cdot \left(1 - \frac{n-C_1}{C_2-C_1}\right) + T_2 \cdot \frac{n-C_1}{C_2-C_1} & C_1 \le n < C_2, \\ T_2 & n \ge C_2, \end{cases}$$

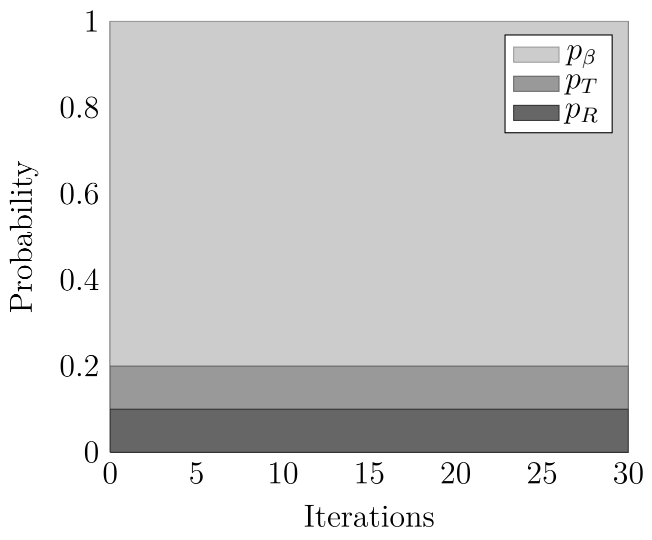
where T_0 corresponds to the value `timeMutator0`, T_1 to the value `timeMutator1`,

T_2 to the value `timeMutator2`, C_1 is the number corresponding to the iteration number of the first spline change `firstStrategyChangeAtIteration` and C_2 to the value `secondStrategyChangeAtIteration`.

3.1.2 Main settings

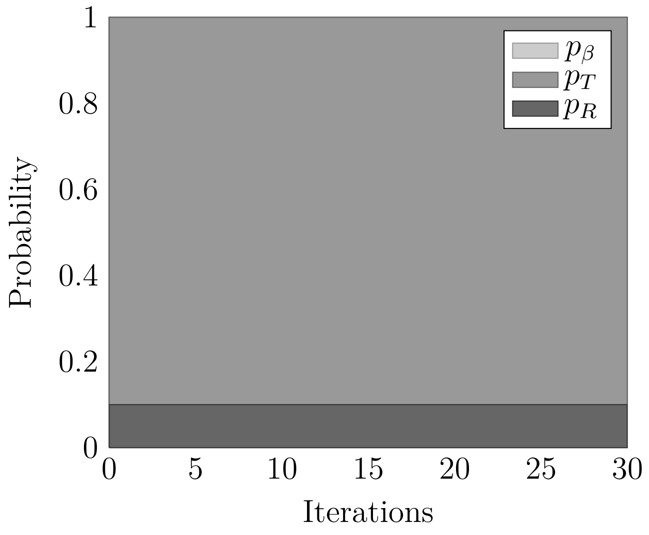
Usually, the values for the probabilities are 0.1 for the ReRouting module, 0.1 for the TimeMutator module and 0.8 for a selection module like SelectExpBeta or BestScore.

Figure 3: Usual settings



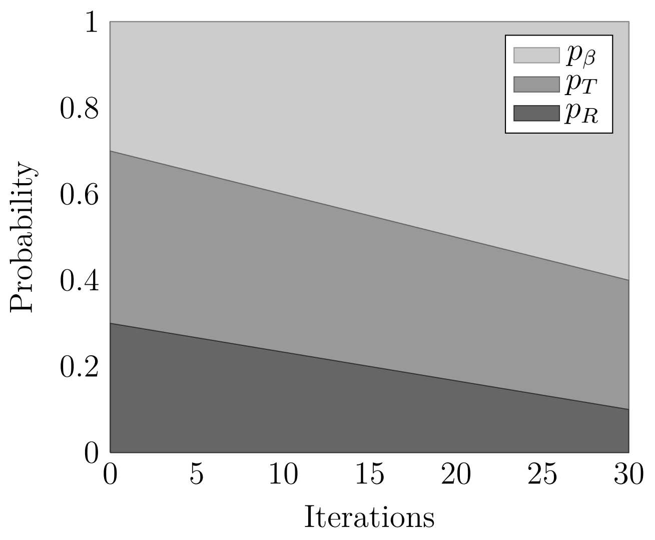
In [Apt10] there is an experimental proof that a value $p_\beta = 0$ leads to a faster convergence. After his analysis for the module SelectExpBeta, he showed that the probability for the rerouting should be small (approximatively $p_R = 0.1$) and that the TimeMutator module can have a probability $p_T = 0.9$.

Figure 4: Optimal constant settings from [Apt10]



The idea for the new strategy module is that perhaps the agents need some module more than the other at the beginning whereas at the end the other modules increase better the score. The first tested configuration is a guess that the module SelectExpBeta is more useful at the end of the run of MATSim than at the beginning.

Figure 5: First idea for iteration-dependent probabilities



The output of the Berlin scenario with this configuration did result in a lower score in comparison with the one obtained previously but converged much faster than the usual settings. As mentioned in [Apt10], the Select-ExpBeta module is almost useless in order to increase the convergence speed and the TimeMutator module should have the largest share. In order to estimate how the average score varies with the different probabilities, an experimental design will be tested and will lead to a regression model for the value of the average score.

3.2 Experimental design

The simulation are based on the Berlin scenario reduced to 1% if its population and 2% of its flow capacities. It needs approximately 4 hours on a private computer[1] for 100 iterations. Therefore heuristics such as genetic algorithms are a bad choice for the optimization of a parameter since these

[1]Dual-core processor T4200 2.0 GHz, 4GB RAM

algorithms need a large sample size and number of iterations to produce a satisfactory result. Experimental designs are very well adapted for the problems which cost much in time and computation power. The technique of the experimental design was originally used in agriculture since it takes sometimes one year before an experiment gives the results (plants breeding). The history and analysis of experimental design can be seen in [GG76].

3.2.1 Configuration

- R_i for $i = 0, 1, 2$ represent the probability that the strategy module choose the ReRouting strategy at iteration C_i, for the other iterations we just use a linear interpolation between those strategies. We have the bounds $0 \leq R_i \leq 1$.
- M_i for $i = 0, 1, 2$ represent the probability that the strategy module choose the TimeMutator strategy at iteration C_i, for the other iterations we just use a linear interpolation between those strategies. We have the bounds $0 \leq M_i \leq 1 - R_i$.
- $C_0 = 0$ is defined implicitly, C_1 and C_2 are the number of iterations for the first and second changes in the strategy module. We have the bounds $C_0 = 0 \leq C_1 \leq C_2 \leq 100$ if the maximum number of iterations is fixed to 100.

The experimental design was set so that it fulfils the conditions $C_1 < C_2$, and $R_i + M_i \leq 1$ for $i = 0, 1, 2$. As a first guess to optimize the speed of convergence, the range of values are near the usually taken values, i.e. $p_T = 0.1$ and $p_R = 0.1$.

As an estimate for the convergence speed, we will define f_{30} to be the average of the average value of the plans of all agents at iteration 30. This can be seen as a good reference for a compromise between convergence speed and the maximum score obtained at equilibrium, since the aim of the optimization is to have a faster convergence but without a significant decrease in the maximum score, otherwise the optimized parameters would not be helpful.

3.2.2 Results

The 41 configurations of the orthogonal experiment were constructed so that the values respect the bounds given above. The same table 1 contains the values of f_{30} of the output of MATSim, and the error of the first linear regression.

3.2.3 Regression and Maxima

Linear Model

The first investigated model for a regression assumes that all parameters are independent, i.e. there is no interaction terms, and the model should be simple. Therefore we want to find a function f which approximates f_{30} using a linear model:

$$f = a_0 + a_1 \cdot R_0 + a_2 \cdot R_1 + a_3 \cdot R_2 + a_4 \cdot M_0 + a_5 \cdot M_1 + a_6 \cdot M_2 + a7 \cdot C_1 + a_8 \cdot C_2.$$

For further convenience we will write

$$\vec{x} = (1, R_0, R_1, R_2, M_0, M_1, M_2, C_1, C_2),$$

and

$$\vec{a} = (a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8).$$

Therefore our model becomes

$$f = \vec{a}\vec{x}^T.$$

Table 1: Configuration and results of the experimental design

R_0	R_1	R_2	M_0	M_1	M_2	C_1	C_2	f_{30}	linear error
0.01	0.01	0.01	0.01	0.01	0.01	10	30	94.1729	-14.098
0.01	0.01	0.10	0.05	0.01	0.05	10	50	107.9107	-2.151
0.01	0.01	0.10	0.05	0.05	0.05	20	30	117.6497	4.307
0.01	0.01	0.01	0.10	0.05	0.01	10	70	118.6562	6.043
0.01	0.05	0.10	0.30	0.05	0.20	20	90	125.8488	-0.440
0.01	0.05	0.20	0.05	0.10	0.01	30	70	121.1412	6.556
0.01	0.05	0.05	0.01	0.01	0.05	30	90	100.4642	-8.917
0.01	0.10	0.10	0.20	0.20	0.20	10	50	128.0114	-1.847
0.01	0.10	0.05	0.05	0.10	0.10	50	70	119.8302	0.423
0.01	0.20	0.05	0.01	0.30	0.10	20	50	127.7462	1.467
0.01	0.20	0.01	0.01	0.20	0.10	20	70	126.5260	5.250
0.01	0.30	0.20	0.10	0.30	0.05	30	50	127.4280	-3.884
0.05	0.01	0.01	0.01	0.05	0.10	30	50	112.1929	-1.604
0.05	0.01	0.10	0.05	0.30	0.10	30	90	126.1508	1.148
0.05	0.01	0.05	0.05	0.20	0.05	20	30	124.8817	4.087
0.05	0.05	0.05	0.01	0.01	0.05	10	70	106.7470	-2.395
0.05	0.05	0.10	0.10	0.10	0.10	10	50	125.2134	6.459
0.05	0.05	0.20	0.30	0.20	0.10	10	30	127.7564	-4.658
0.05	0.10	0.01	0.10	0.05	0.05	20	70	122.9677	6.368
0.05	0.10	0.20	0.01	0.01	0.01	20	90	105.2245	-3.712
0.05	0.20	0.10	0.30	0.05	0.01	30	70	127.5707	0.769
0.05	0.30	0.05	0.05	0.01	0.20	10	70	115.9650	-2.451
0.05	0.30	0.05	0.20	0.05	0.01	20	50	126.4087	2.388
0.10	0.01	0.05	0.30	0.30	0.01	10	70	127.3280	-6.675
0.10	0.05	0.01	0.05	0.01	0.01	20	50	113.8343	1.288
0.10	0.05	0.10	0.10	0.01	0.10	20	30	121.4590	4.197
0.10	0.20	0.01	0.20	0.10	0.05	10	90	127.0290	2.758
0.10	0.30	0.10	0.01	0.20	0.05	30	70	124.2512	0.073
0.20	0.01	0.05	0.30	0.10	0.05	20	50	128.0398	-1.473
0.20	0.01	0.20	0.20	0.01	0.10	20	70	125.3030	3.995
0.20	0.05	0.01	0.05	0.20	0.01	30	50	126.7749	3.146
0.20	0.05	0.10	0.01	0.05	0.05	50	70	111.8906	-5.099
0.20	0.10	0.10	0.01	0.30	0.01	10	30	127.6955	1.764
0.20	0.30	0.01	0.05	0.05	0.10	10	90	119.7026	-0.459
0.30	0.01	0.05	0.10	0.20	0.01	50	90	127.4458	0.169
0.30	0.01	0.10	0.20	0.01	0.10	30	70	126.7333	2.501
0.30	0.05	0.05	0.01	0.05	0.10	10	50	120.3244	2.900
0.30	0.05	0.01	0.05	0.30	0.20	20	70	128.9096	-3.74
0.30	0.10	0.01	0.30	0.01	0.05	30	50	128.3012	-1.555
0.30	0.20	0.20	0.05	0.05	0.05	10	30	120.4604	-0.480
0.30	0.30	0.10	0.01	0.10	0.01	20	30	120.7744	-2.413

Let Y be the vector of all the experimental results for the average of the average plan value for all users, and let X be the matrices which contains all configurations, that is each line correspond to one vector $\vec{x}$. A least-square regression gives us

$$\hat{a} = (X^T X)^{-1} \cdot X^T Y.$$

We obtain the following values for $\hat{a}$:

$$\hat{a} = \begin{pmatrix} 107.02 & 18.88 & 16.63 & -1.97 & 43.88 & 43.98 & 22.412 & 0.08 & -0.03 \end{pmatrix},$$

with $R^2 = 0.73315$, and the following t-stats

$$\hat{p} = \begin{pmatrix} 33.13 & 2.58 & 2.17 & -0.16 & 5.66 & 5.80 & 1.58 & 1.11 & -0.81 \end{pmatrix}.$$

With a confidence interval of 95% the estimated parameters for a_3, a_6, a_7 and a_8 are not significantly different from 0. By removing these parameters the estimated parameters are $(a_0, a_1, a_2, a_4, a_5) = (107.89, 19.99, 15.73, 43.71, 45.98)$ and $R^2 = 0.71$, since R^2 is not near 1 we will investigate another model.

Quadratic model

Our new model for f is:

$$\begin{aligned} f &= a + b_0R_0 + b_1R_1 + b_2R_2 + c_0M_0 + c_1M_1 + c_2M_2 + d_1C_1 + d_2C_2 \\ &\quad e_0R_0^2 + e_1R_1^2 + e_2R_2^2 + f_0M_0^2 + f_1M_1^2 + f_2M_2^2 + g_1C_1^2 + g_2C_2^2 \\ &\quad +h_0R_0C_1 + h_1R_1C_1 + h_2R_2C_1 + i_0M_0C_1 + i_1M_1C_1 + i_2M_2C_1 \\ &\quad +j_0R_0C_2 + j_1R_1C_2 + j_2R_2C_2 + k_0M_0C_2 + k_1M_1C_2 + k_2M_2C_2. \end{aligned}$$

We want to express $f = \frac{1}{2}x^T Hx + c^T x + \alpha$. we have $\alpha = a$ and

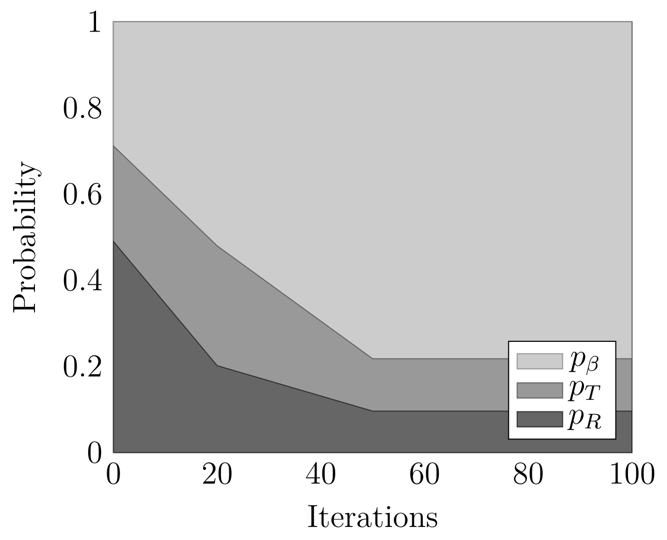
$$H = \begin{pmatrix} 2e_0 & 0 & 0 & 0 & 0 & 0 & h_01 & j_0 \\ 0 & 2e_1 & 0 & 0 & 0 & 0 & h_1 & j_1 \\ 0 & 0 & 2e_2 & 0 & 0 & 0 & h_2 & j_2 \\ 0 & 0 & 0 & 2f_0 & 0 & 0 & i_0 & k_0 \\ 0 & 0 & 0 & 0 & 2f_1 & 0 & i_1 & k_1 \\ 0 & 0 & 0 & 0 & 0 & 2f_2 & i_2 & k_2 \\ h_0 & h_1 & h_2 & i_0 & i_1 & i_2 & 2g_1 & 0 \\ j_0 & j_1 & j_2 & k_0 & k_1 & k_2 & 0 & 2g_2 \end{pmatrix}, \qquad c = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ c_0 \\ c_1 \\ c_2 \\ d_1 \\ d_2 \end{pmatrix}.$$

One obtains $a = 82.209$, $b_0 = 59.75$, $b_1 = 57.85$, $b_2 = 57.68$, $c_0 = 62.24$, $c_1 = 153.74$, $c_2 = 33.39$, $d_1 = 0.31$, $d_2 = 0.46$, $e_0 = -115.93$, $e_1 = -89.13$, $e_2 = -149.67$, $f_0 = -278.71$, $f_1 = -311.50$, $f_2 = -218.14$, $g_1 = -0.008$, $g_2 = -0.003$, $h_0 = 0.23$, $h_1 = -1.26$, $h_2 = 0.16$, $i_0 = 2.18$, $i_1 = -0.42$, $i_2 = 1.30$, $j_0 = -0.26$, $j_1 = 0.037$, $j_2 = -0.32$, $k_0 = 0.24$, $k_1 = 0.035$, $k_2 = 0.024$, with $R^2 = 0.936$.

Using t-stats, all parameters that depends on C_1 or C_2 are not significantly different from 0. By removing these variables one obtain the following values: $a = 98.16$, $b_0 = 30.94$, $b_1 = 57.43$, $b_2 = 35.37$, $c_0 = 130.87$, $c_1 = 100.73$, $c_2 = 63.63$, $e_0 = -31.56$, $e_1 = -142.30$, $e_2 = -183.84$, $f_0 = -294.93$, $f_1 = -181.06$, $f_2 = -261.70$ and $R^2 = 0.854$.

Using $\nabla f = 0$ we obtain the optimal parameters: $R_0 = 0.49$, $R_1 = 0.20$, $R_2 = 0.096$, $T_0 = 0.22$, $T_1 = 0.28$, $T_2 = 0.12$. Since C_1 and C_2 were not estimated by this technique, they will not interact much with the average of the average score at iteration 30. For the plot the values $C_1 = 20$ and $C_2 = 50$ are used.

Figure 6: Optimal results



3.2.4 Discussion of the regression

Since the regression concerns only the value f_{30}, i.e. the value of the average score at iteration 30, this value should not depend on the probability of each strategy module after that iteration, therefore is it clear that there are too many explanatory variables in the model. Hence the new model that will be investigated should only depend on 4 parameters: R_0, R_{30}, T_0 and T_{30} such that

$$p_T(n) = \begin{cases} T_0 \cdot \left(1 - \frac{n}{30}\right) + T_{30} \cdot \frac{n}{30} & n < 30, \\ T_{30} & n \geq 30, \end{cases}$$

and

$$p_R(n) = \begin{cases} R_0 \cdot \left(1 - \frac{n}{30}\right) + R_{30} \cdot \frac{n}{30} & n < 30, \\ R_{30} & n \geq 30. \end{cases}$$

Moreover there was a statistical problem in the previous approach: the regression's aim was to determine 29 parameters out of only 41 observations.

Such a regression cannot be very representative of the real parameters. After the removal of the non significant parameters, there is still 13 parameters to be estimated with 41 observations.

Another technique was used instead of a multi-linear regression, which is described in [JSW98], but this technique has the same statistical problem as the linear regression (too small sample for a large number of parameters).

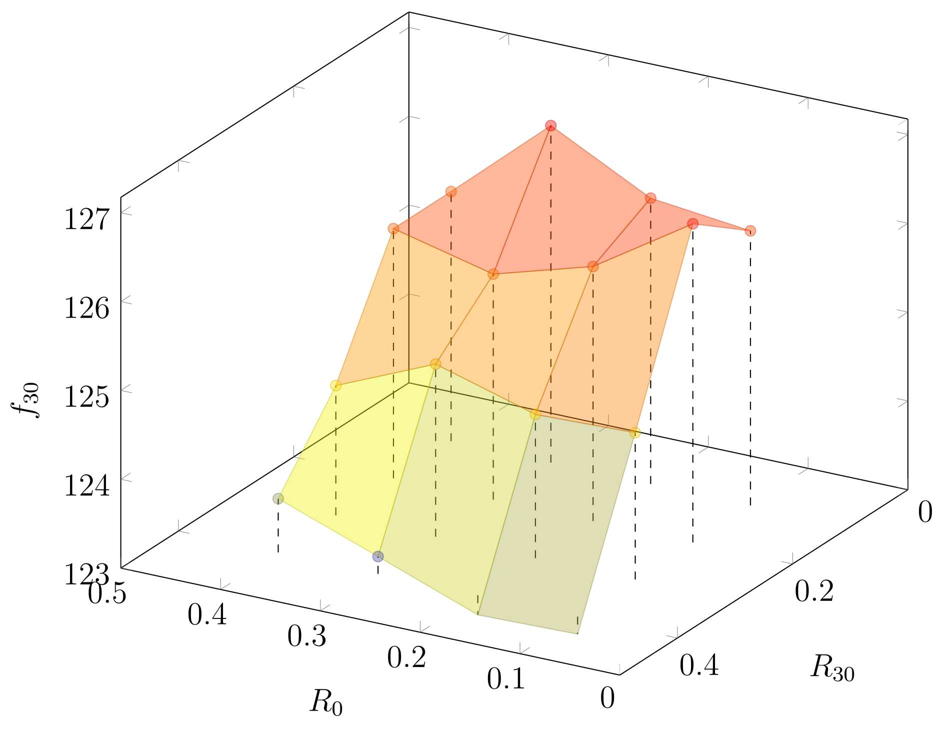
3.3 Final model

3.3.1 Normal scenario

Since the linear regression did not provide significant results, it would be interesting to have a better view on how f_{30} varies with the parameters. Therefore some configurations are made using a grid of value for the parameters.

- The first grid is for $(R_0, R_{30}) \in \{0.1, 0.2, 0.3, 0.4\} \times \{0.1, 0.2, 0.3, 0.4\}$, and $T_0 = T_{30} = 0.1$ fixed. Each combination of these parameters is evaluated by MATSim, the average score at iteration 30 is plotted in the next figure.

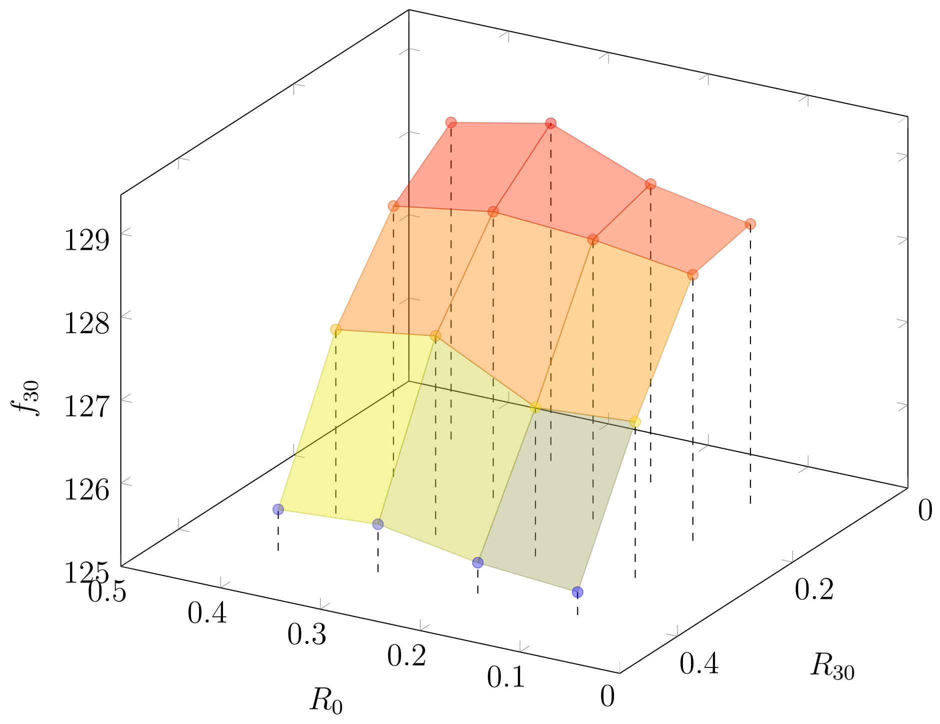
Figure 7: f_{30} for $T_0 = T_{30} = 0.1$ fixed



First of all it is reassuring that the function seems to be smooth enough and not a random function. The picture shows that f_{30} is the larger when R_{30} becomes small, but R_0 can be non-zero, in fact the maximum in the graph is attained for $R_0 = 0.3$. It is important to notice that the simulation with MATSim relies on randomness and each run with the same parameters should output a different results, but in MATSim the random number are generated thanks to a random seed set in the configuration file, therefore the same sequence of random numbers will be created for every simulation. The surface composed of the different values of f_{30} is similar to a parabolic surface in the direction of R_{30}, which indicates that the linear regression with some squared variables might output a good estimation.

- Sample for $(R_0, R_{30}) \in \{0.1, 0.2, 0.3, 0.4\} \times \{0.1, 0.2, 0.3, 0.4\}$, and $T_0 = T_{30} = 0.2$ fixed. It is hard to plot a graph depending on more than 2 parameters and each run requires 8 hours of simulation on a computer, therefore a sampling grid on more than 2 parameters at a time is not convenient.

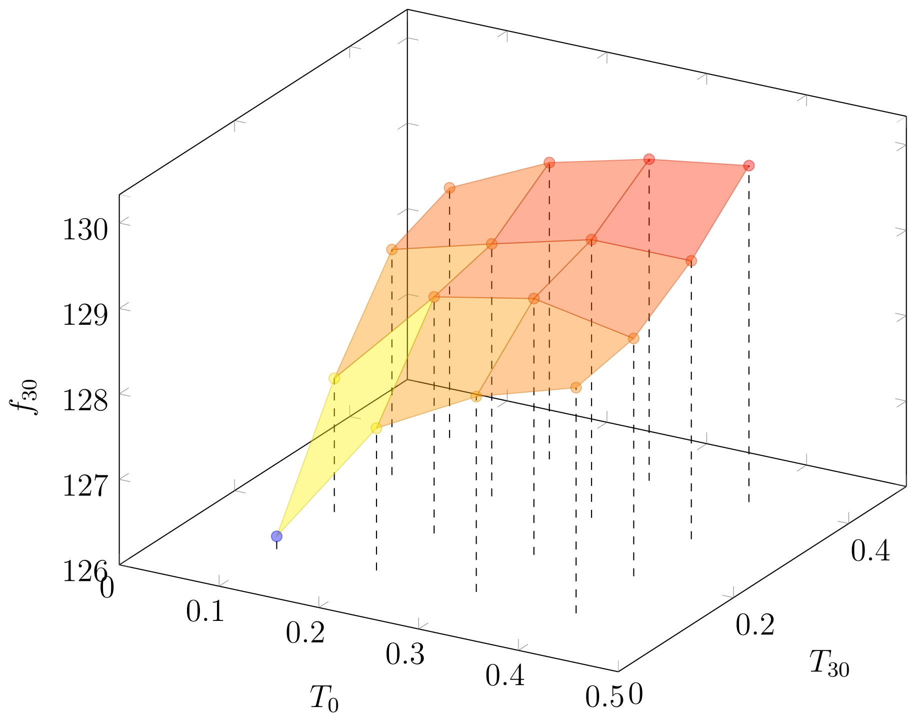
Figure 8: f_{30} for $T_0 = T_{30} = 0.2$ fixed



The results for this configurations corroborates the fact that f_{30} can be described by a quadratic function since the shape of the graph for $T_0 = T_{30} = 0.1$ and for $T_0 = T_{30} = 0.2$ have the same aspect, except a little shift up of the score for the second graph.

- Sample for $(T_0, T_{30}) \in \{0.1, 0.2, 0.3, 0.4\} \times \{0.1, 0.2, 0.3, 0.4\}$, and $R_0 = R_{30} = 0.1$ fixed. The opposite configurations are also simulated with MATSim, this time the ReRouting has a fix probability and the TimeMutator probabilities are sampled on a grid. The values of the probabilities for the SelectExpBeta module are always expressed implicitly as the complementary of the sum of the two other probabilities so that the total sum is always equal to one.

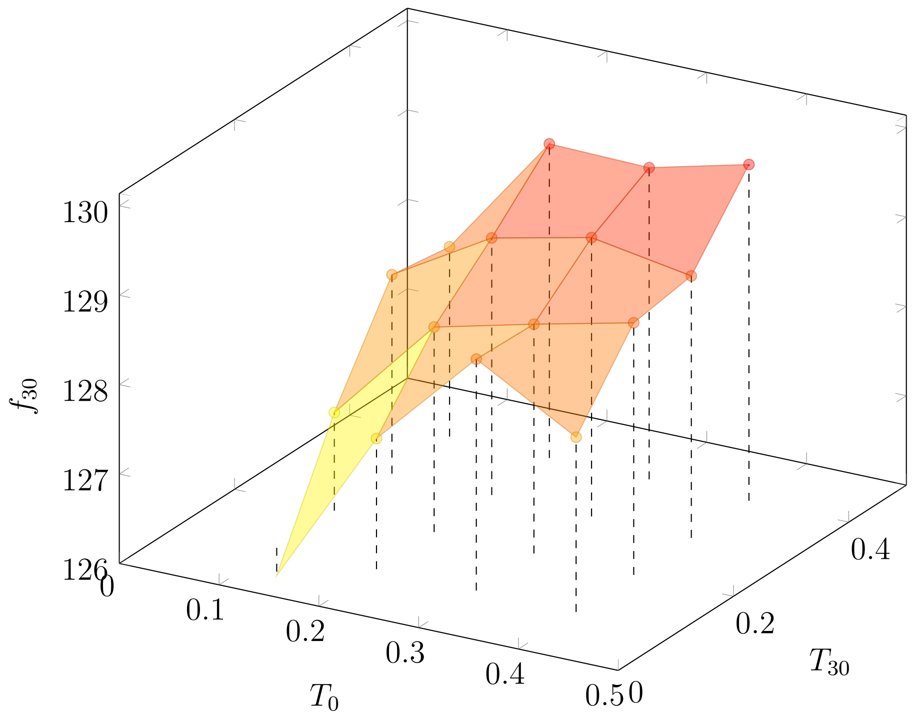
Figure 9: f_{30} for $R_0 = R_{30} = 0.1$ fixed



The orientation of the graph for this new sampling looks also like a quadratic function, but the general direction is in diagonal, hence both values for T_0 and T_{30} needs to be large.

- Sample for $(T_0, T_{30}) \in \{0.1, 0.2, 0.3, 0.4\} \times \{0.1, 0.2, 0.3, 0.4\}$, and $R_0 = R_{30} = 0.2$ fixed.

Figure 10: f_{30} for $R_0 = R_{30} = 0.2$ fixed



The same conclusion as for the previous sampling holds here: the probabilities for the TimeMutator strategy needs to be large in all iterations.

Since the curves seems smooth enough, a new attempt is made for the regression. There is only 4 variables, but we add quadratic terms since the curves do not resemble a plane. The model is then given by:

$$f_{30} = a_1 + a_2 \cdot R_0 + a_3 \cdot R_{30} + a_4 \cdot T_0 + a_5 \cdot T_{30} + a_6 \cdot R_0^2 + a_7 \cdot R_{30}^2 + a_8 \cdot T_0^2 + a_9 \cdot T_{30}^2.$$

The least-square regression gives the following estimated parameters:

$$a = (122.63, 2.92, 5.51, 22.82, 14.94, -5.29, -33.28, -34.86, -16.61)\,,$$

with $R^2 = 0.9709$ which indicates that these parameters fits well the data. As one can see the parameters corresponding to the quadratic terms are all negative, therefore we can find a maximum by setting the gradient to 0: $\nabla f_{30} = 0$, then

$$\begin{aligned} R_0 &= 0.2757 \\ R_{30} &= 0.0827 \\ T_0 &= 0.3273 \\ T_{30} &= 0.4498 \end{aligned}$$

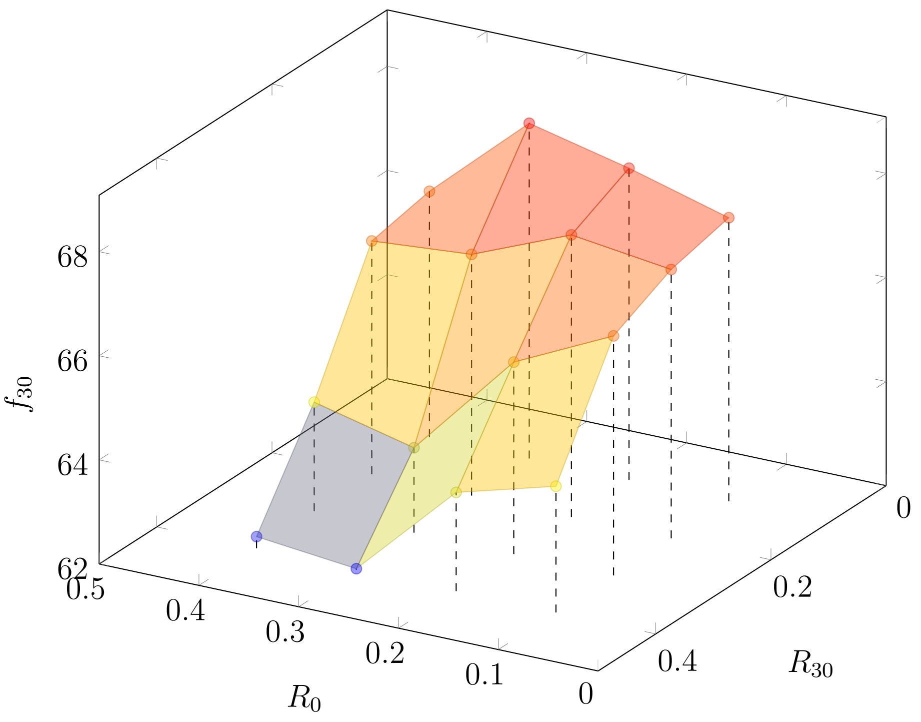
Since the optimal variables satisfy the bounds, one can calculate the theoretical optimal value $f_{30} = 130.3560$, and run a simulation in order to obtain the practical value, which is $f_{30} = 130.00113$. The model predicts well enough the optimal value. It is interesting to see how the score function would depend on the congestion of the scenario and if the optimal variables are the same in all cases.

3.3.2 Congested Scenario

The same 4 samples are run on a more congested scenario where the flow capacity of each link capacity is divided by 4.

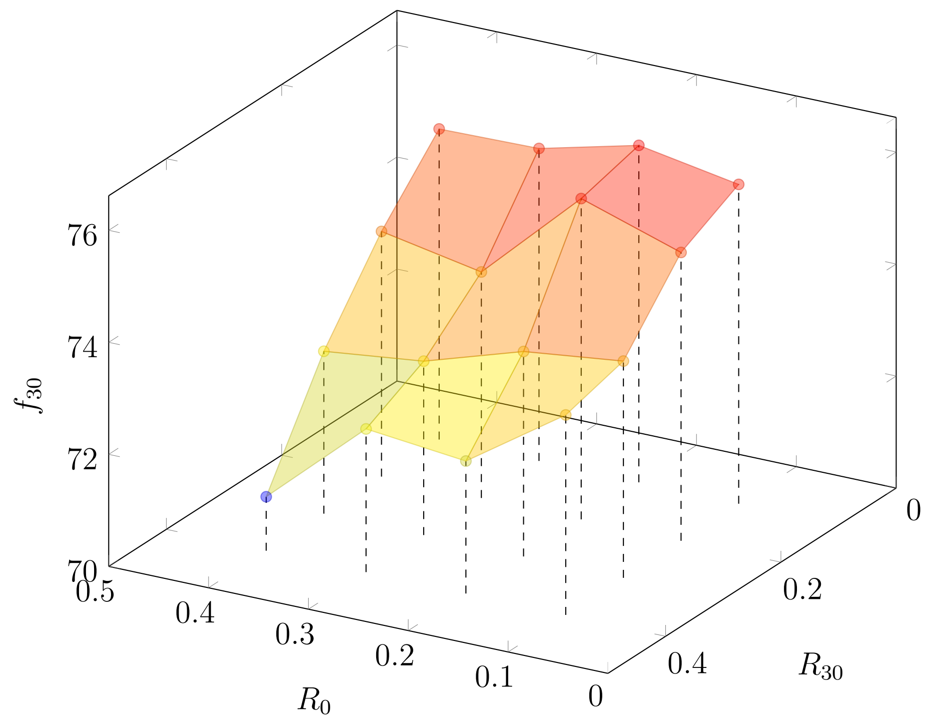
- The first grid is for $(R_0, R_{30}) \in \{0.1, 0.2, 0.3, 0.4\} \times \{0.1, 0.2, 0.3, 0.4\}$, and $T_0 = T_{30} = 0.1$ fixed.

Figure 11: f_{30} for $T_0 = T_{30} = 0.1$ fixed



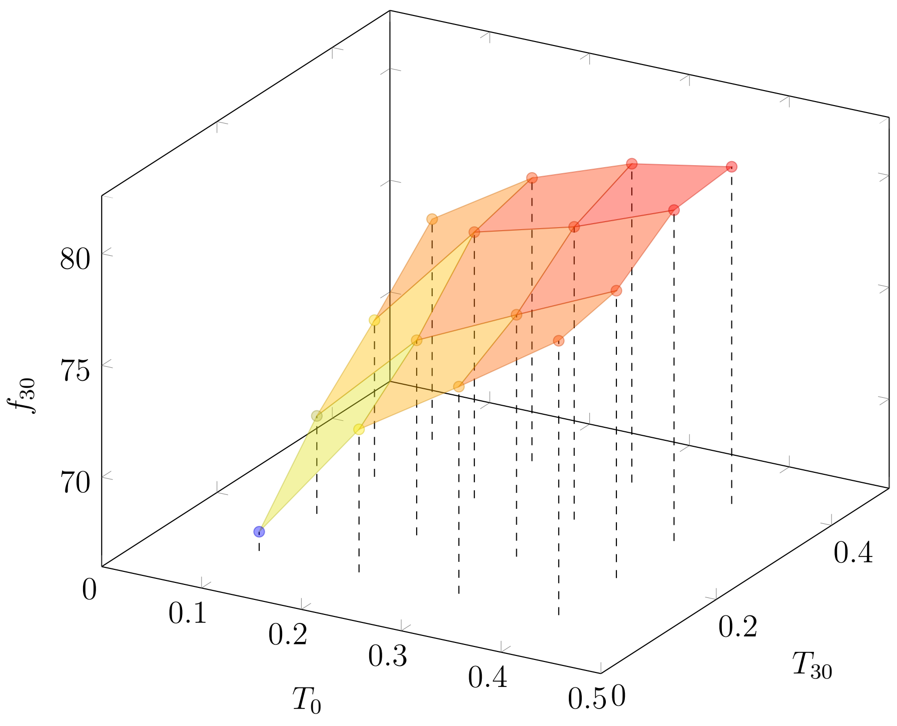
- Sample for $(R_0, R_{30}) \in \{0.1, 0.2, 0.3, 0.4\} \times \{0.1, 0.2, 0.3, 0.4\}$, and $T_0 = T_{30} = 0.2$ fixed.

Figure 12: f_{30} for $T_0 = T_{30} = 0.2$ fixed



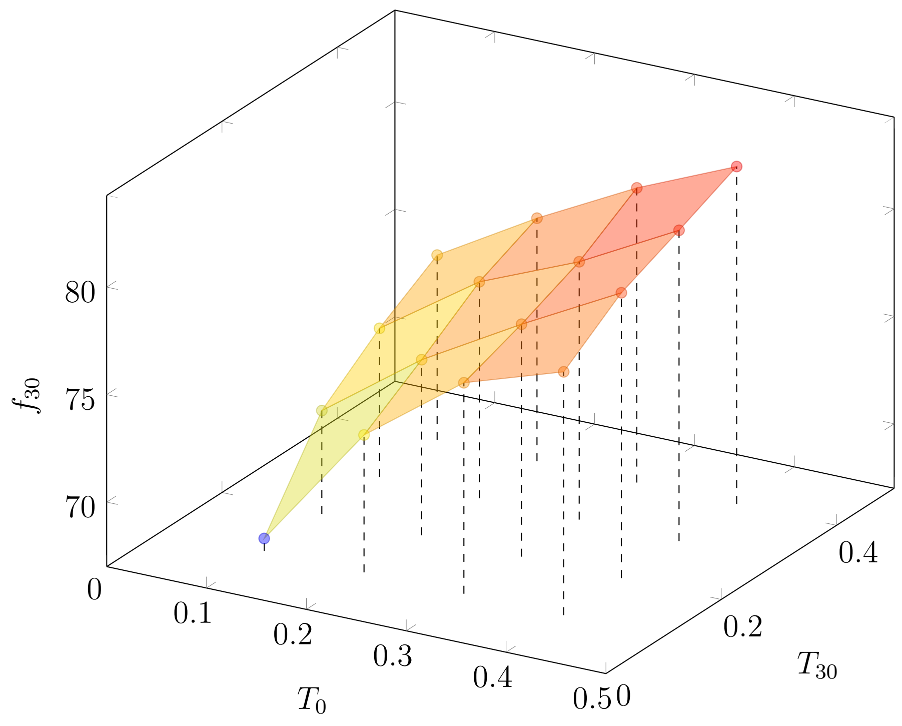
- Sample for $(T_0, T_{30}) \in \{0.1, 0.2, 0.3, 0.4\} \times \{0.1, 0.2, 0.3, 0.4\}$, and $R_0 = R_{30} = 0.1$ fixed.

Figure 13: f_{30} for $R_0 = R_{30} = 0.1$ fixed



- Sample for $(T_0, T_{30}) \in \{0.1, 0.2, 0.3, 0.4\} \times \{0.1, 0.2, 0.3, 0.4\}$, and $R_0 = R_{30} = 0.2$ fixed.

Figure 14: f_{30} for $R_0 = R_{30} = 0.2$ fixed



As in the normal scenario, the curves for each sample are quite smooth. The same model is used for the regression:

$$f_{30} = a_1 + a_2 \cdot R_0 + a_3 \cdot R_{30} + a_4 \cdot T_0 + a_5 \cdot T_{30} + a_6 \cdot R_0^2 + a_7 \cdot R_{30}^2 + a_8 \cdot T_0^2 + a_9 \cdot T_{30}^2.$$

The least-square regression gives the following estimated parameters:

$$a = (55.63, 13.75, 10.67, 71.97, 46.70, -35.45, -46.56, -85.87, -52.63).$$

with $R^2 = 0.9701$ which indicates that these parameters also fits well the data. As one can see the parameters corresponding to the quadratic terms are all negative, therefore we can find a maximum by setting the gradient to 0: $\nabla f_{30} = 0$, then

$$\begin{aligned} R_0 &= 0.1940 \\ R_{30} &= 0.1146 \\ T_0 &= 0.4190 \\ T_{30} &= 0.4437 \end{aligned}$$

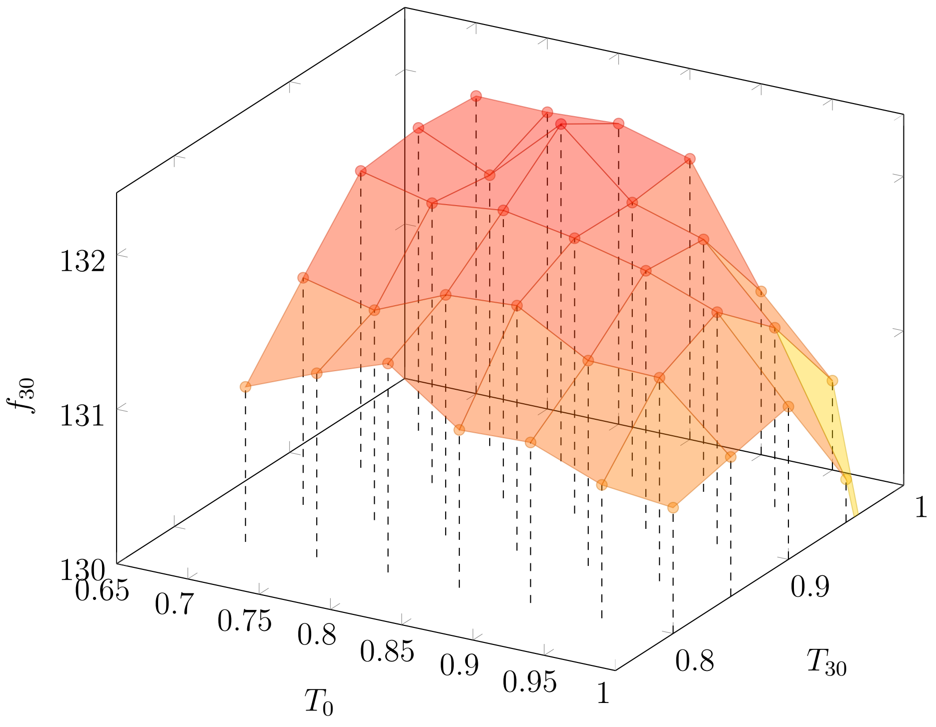
Since the optimal variables satisfy the bounds, one can calculate the theoretical optimal value $f_{30} = 83.0118$, and run a simulation in order to obtain the practical value, which is $f_{30} = 83.4162$. The model predicted well enough the optimal value. The biggest difference for the estimated parameters between the normal and the congested scenario are for the values of the probabilities at iteration 0: in the first case $R_0 = 0.2757$, and in the congested case $R_0 = 0.1940$, the optimal value for R_0 should probably lie between those two values. T_0 has also two different values, 0.3273 in the first case and 0.4190 in the second case. This could be explained by the fact that in a congested scenario the time of the departure plays a big role in the congestion of the network, therefore the algorithm needs more run of TimeMutator to obtain a better results. The values at the iteration 30 are very similar in both scenarios.

3.3.3 Global optimum

The obtained parameters could be satisfactory but the analysis of [Apt10] shows that the optimal constant parameters are $p_T = 0.9$ and $p_R = 0.1$ for the probabilities of selecting either the TimeMutator module or the ReRouting module. The estimated probabilities from the previous model did not reach higher value than 0.5, therefore it is necessary to simulate the same scenario over a sample which deals with higher probabilities for p_T.

- Sample for $(T_0, T_{30}) \in \{0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 1.0\} \times \{0.8, 0.85, 0.9, 0.95, 1\}$, and $R_0 = 1 - T_0$, $R_{30} = 1 - T_{30}$.

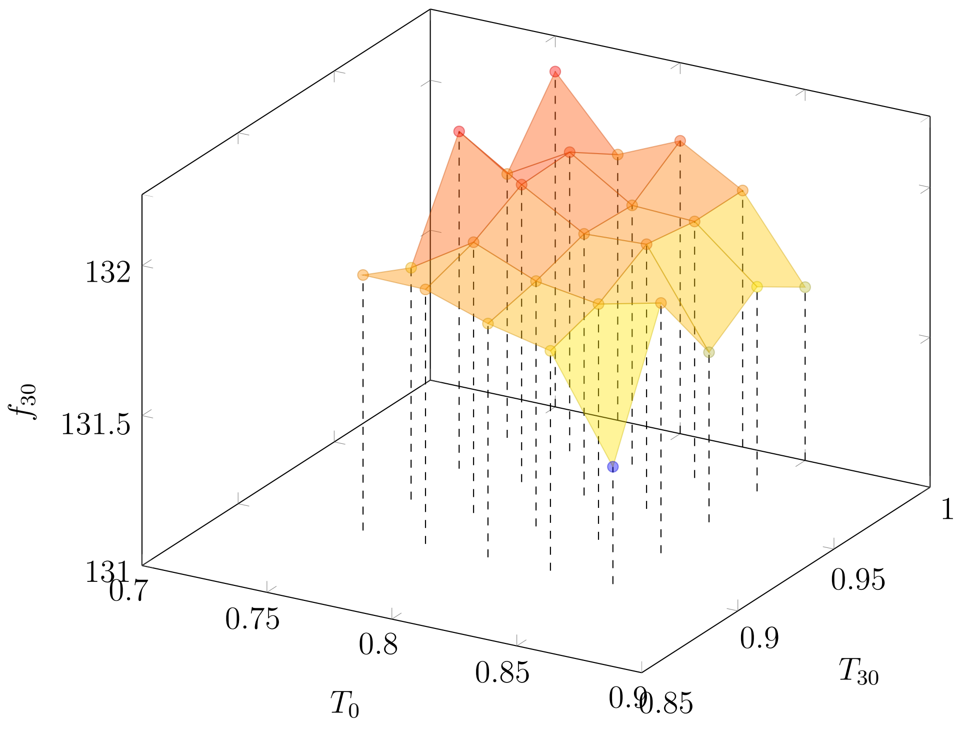
Figure 15: f_{30} for $R_0 = 1 - T_0$ and $R_{30} = 1 - T_{30}$



The graph in the plot looks like a concave function with some random noise, which seems also to contain the maximum. It should be in the interval $(T_0, T_{30}) \in [0.75, 0.85] \times [0.9, 1.0]$, then a sample around these values is created.

- A refinement grid is made for $(T_0, T_{30}) \in \{0.75, 0.775, 0.8, 0.825, 0.85\} \times \{0.9, 0.925, 0.95, 0.975, 1.0\}$, and $R_0 = 1 - T_0$, $R_{30} = 1 - T_{30}$.

Figure 16: f_{30} for $R_0 = 1 - T_0$ and $R_{30} = 1 - T_{30}$



Because of the randomness of the noise it is hard to determine a single values for the optimum. The optimal parameters should be in the interval $(R_0, R_{30}, T_0, T_{30}) \in [0.2, 0.25] \times [0, 0.05] \times [0.75, 0.8] \times [0.95, 1.0]$. The same setting were done on a more congested scenario and the interval for the optimal parameters is approximatively the same.

3.4 Final results

The sampling approach provides an interval for the optimal values of the parameters between the iterations 0 and 30. As we saw, the probability for SelectExpBeta can be very low and even zero in order to increase the speed of convergence. As a small conclusion one will see some simulations of the Berlin scenario with the optimal settings (iteration-dependent probabilities) and some variants, and compare the results with those from the standard settings.

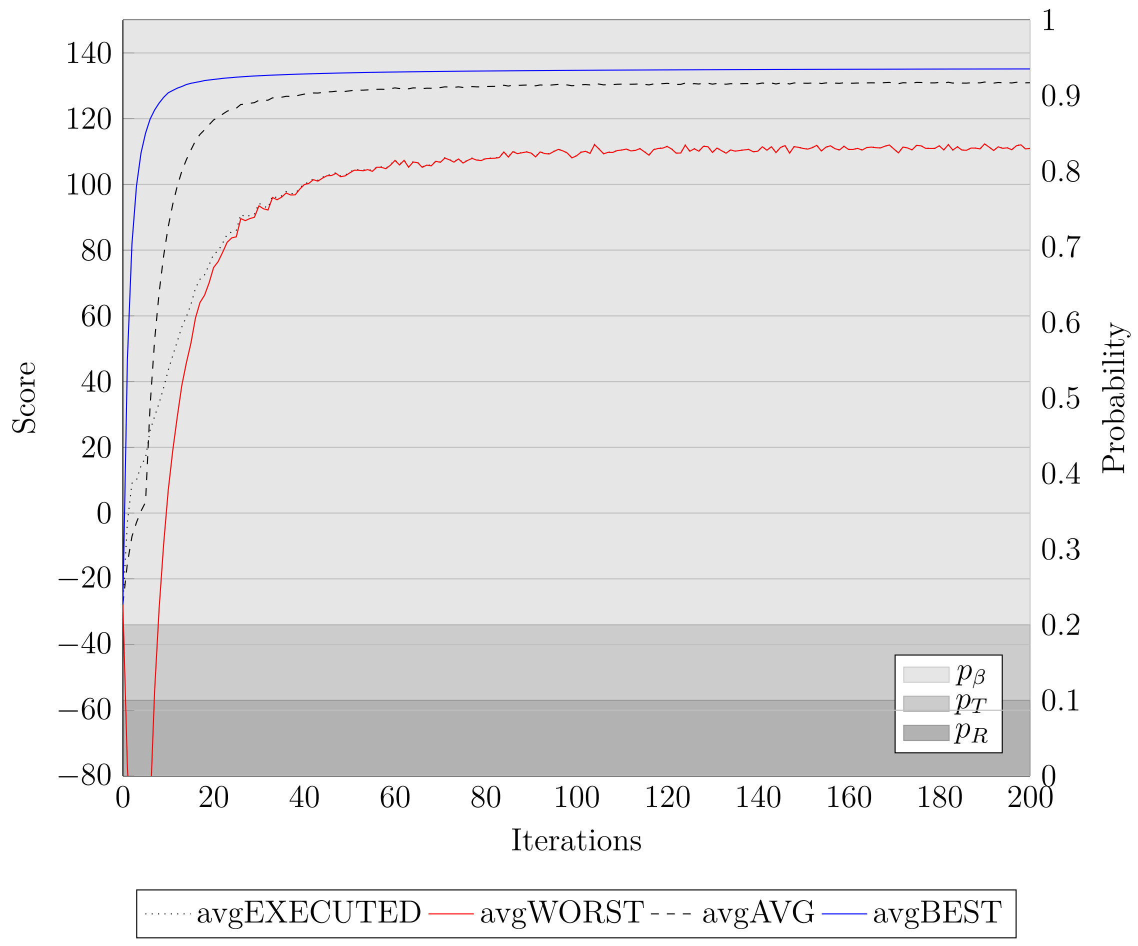
In order to have a good overview of the probabilities which change with the iterations and the corresponding score at each iteration, the scores of each simulation is included in the graph: The axis on the left will describe the scale of the score whereas the axis on the right is simply the repartition of the probability.

3.4.1 Non-congested scenario

We found a range of optimal values for the distribution of probabilities between the iterations 0 and 30, but the probabilities were not studied after the iteration 30, but one can expect that in order to optimize the speed of convergence the probabilities won't change much when we reach the equilibrium.

- The first simulation concerns the scenario with the settings that have always been used in MATSim. These settings assume the need of a great share of probability for the SelectExpBeta. The previous analysis shows that this parameter decreases the speed of convergence, but might have an importance for the stability of the iterative procedure.

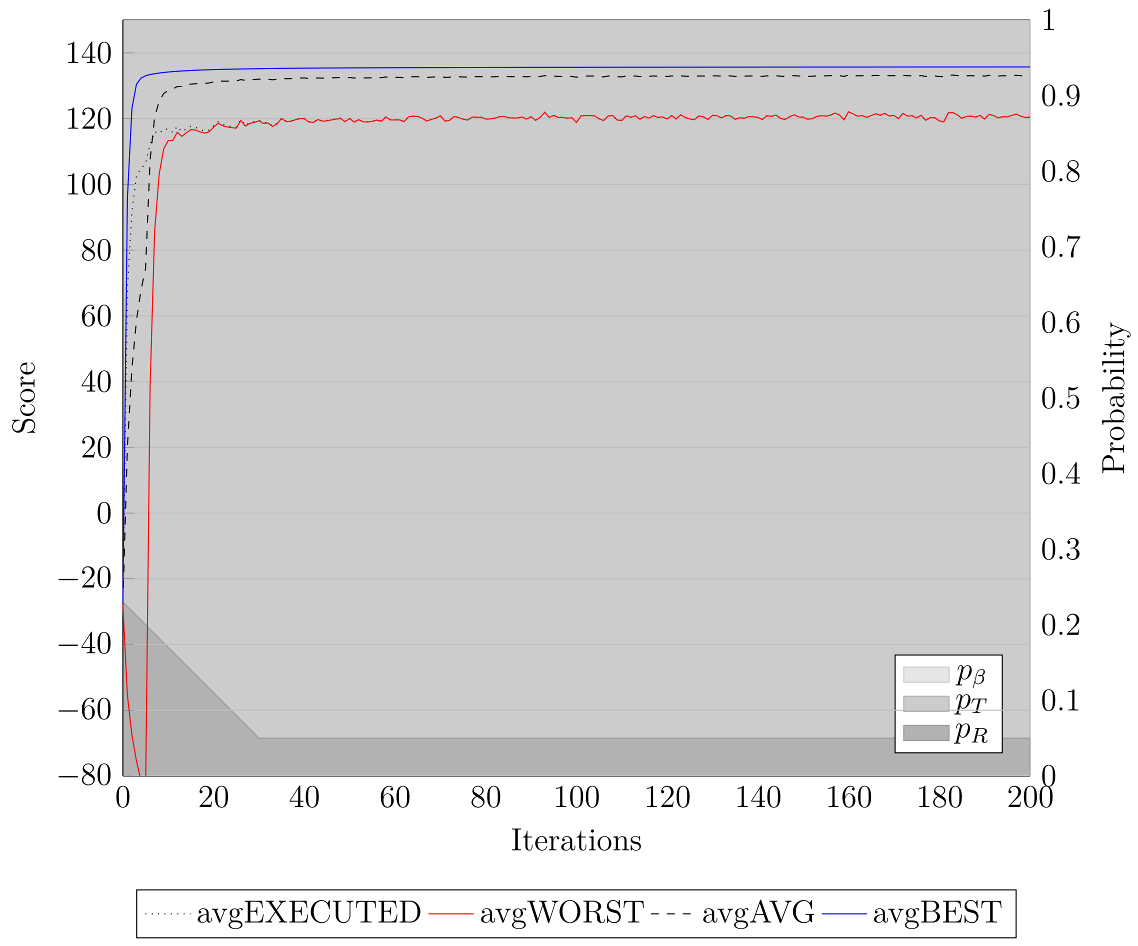
Figure 17: Non-congested scenario: commonly used settings



The average of the best scores after 200 iterations reaches 135.13 and the average of the average scores reaches 131. The average of the average scores exceeds the value of 130 after 84 iterations.

- The first different configuration is simply the calculated optimal settings between the iterations 0 and 30 and the probabilities remains constant after the iteration 30. During all the process, the module SelectExpBeta is never executed.

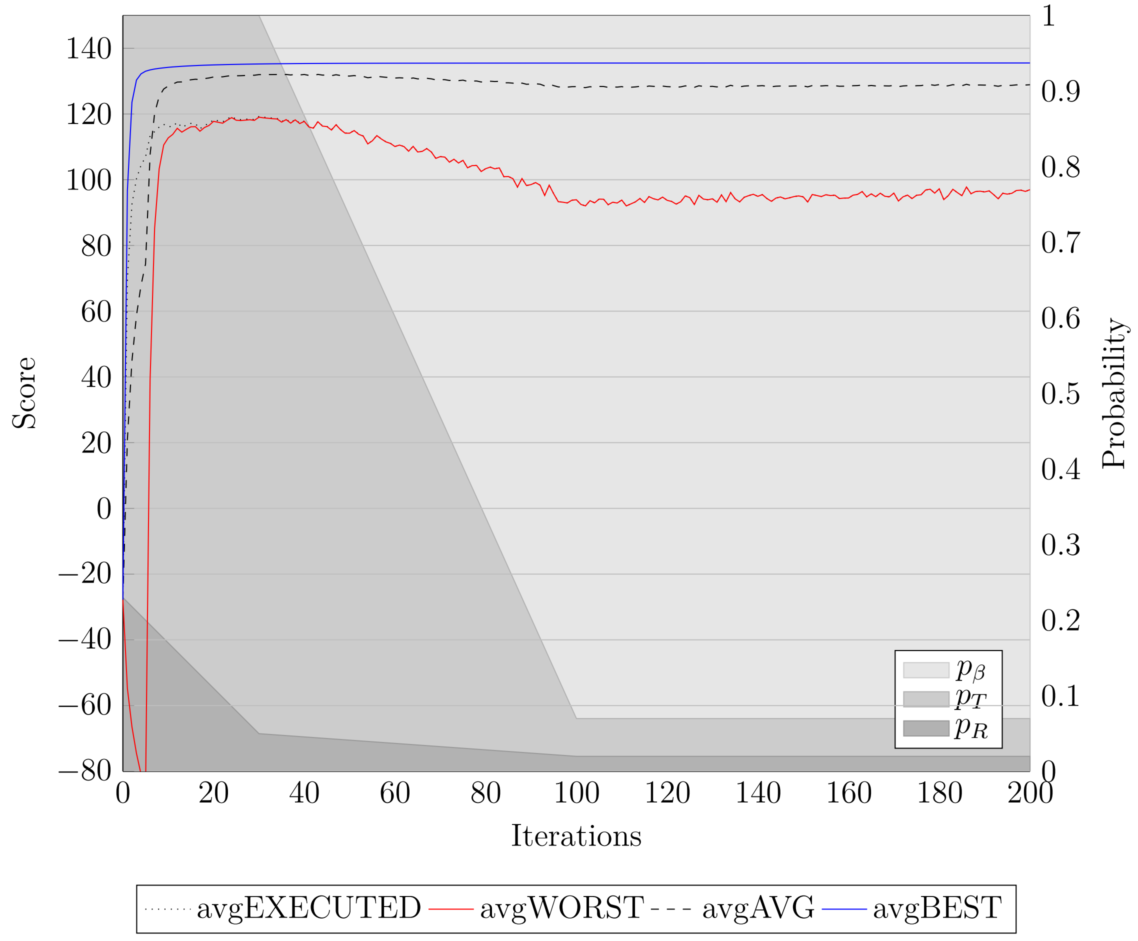
Figure 18: Non-congested scenario: iteration-dependent settings, first variant



In this case, the maximum value for the average of the best scores is 135.74, the average of the average scores is 133.1 and it exceeds the value of 130 after 14 iterations. In comparison to the constant usual settings the convergence is really faster in this case.

- In order to observe what happens if the module SelectExpBeta is executed with a certain probability after iteration 30, a new configuration for the parameters is set as seen in the following picture. This is done since it could be expected that the module SelectExpBeta helps the convergence to an equilibrium.

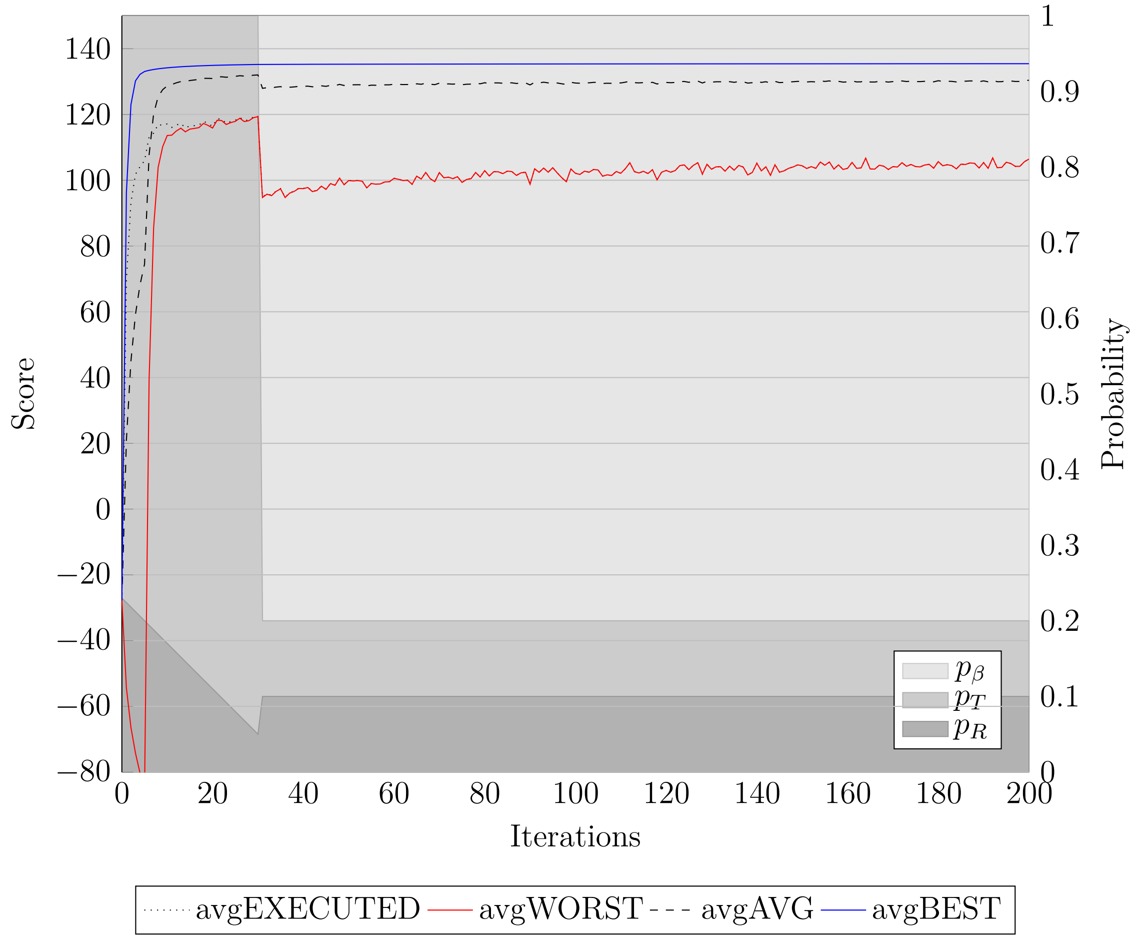
Figure 19: Non-congested scenario: iteration-dependent settings, second variant



Here the average of the best scores is 135.5, the average of the average scores is 128.9. The average of the average scores still exceeds the value of 130 after 14 iterations, but at the moment that the module SelectExp-Beta has a non-zero probability, the score drops out proportionally to the probability for this module. In this case the module SelectExpBeta shifts the average of the average score down by more than a value of 1. Moreover after the add of SelectExpBeta strategy, the average executed score is almost always the same as the average of the worst score, therefore almost each executed plan becomes the worst plan of the agent's memory.

- Instead of doing a progressive introduction of the SelectExpBeta, this configuration shows the results for a sudden shift from the optimal iteration-dependent settings to the constant usual settings.

Figure 20: Non-congested scenario: iteration-dependent settings, third variant



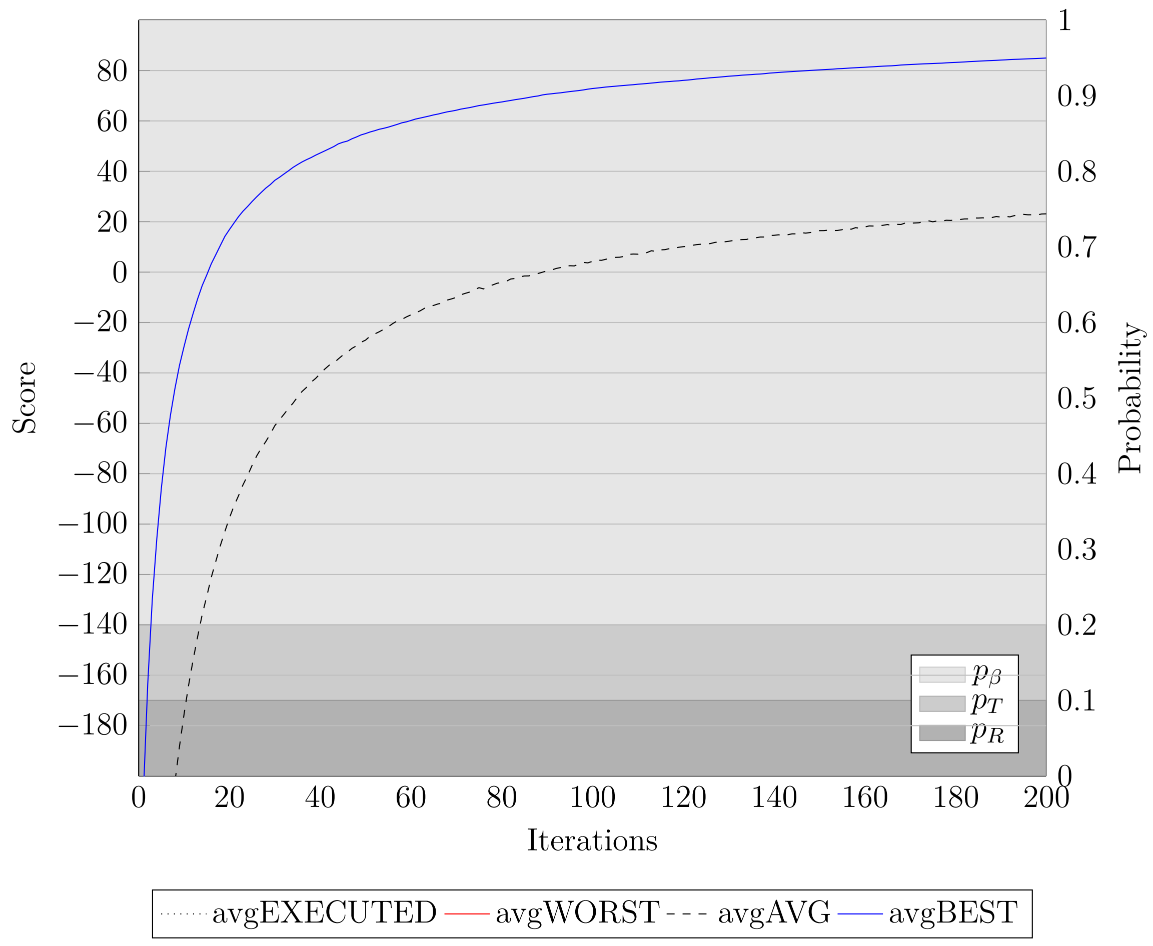
The sudden change of the probabilities causes also a sudden change in the score. The average and worst score are shifted down: the average of the worst score is 119.34 at iteration 30 and 94.74 at iteration 31 after the change of the distribution of probabilities. The average of the average score is 132 at iteration 30 and 127.9 at iteration 31. The final average score reaches 130.2 and the best score 135.4.

3.4.2 Congested

The same tests are run on a very congested scenario in order to see if the iterative-dependent probabilities really improves much the speed of convergence in this case.

- The first configuration is simply the usual settings for MATSim.

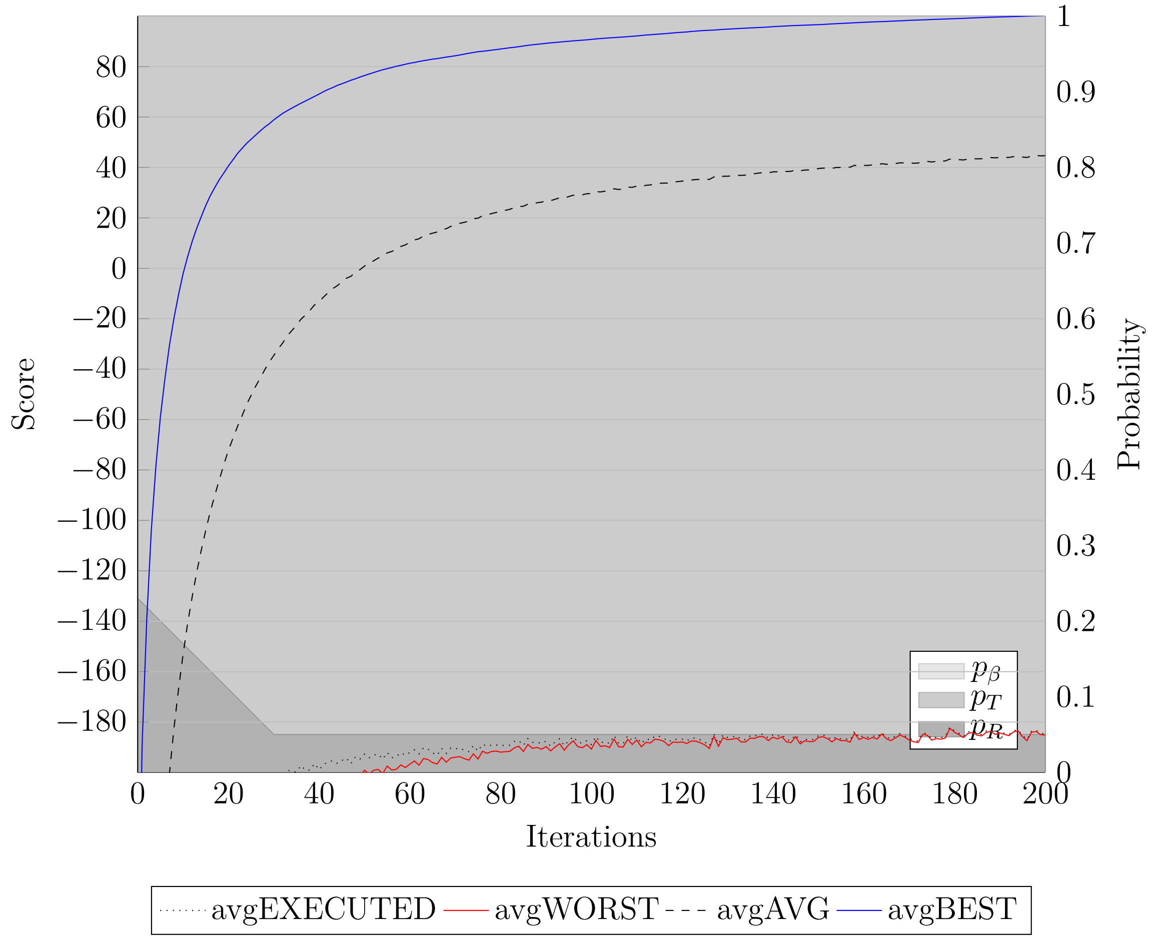
Figure 21: Congested scenario: commonly used settings (Congested case)



The figure shows that 200 iterations is not sufficiently at all in order to reach the equilibrium in a very congested scenario. At iteration 200, the average of the average scores is 23.12, the average of the best scores 84.96, and the average of the average scores exceeds 0 after 90 iterations.

- This configuration consists of the optimal iteration-dependent settings that were found previously, hence there is a probability of zero that the module SelectExpBeta is executed throughout the whole simulation.

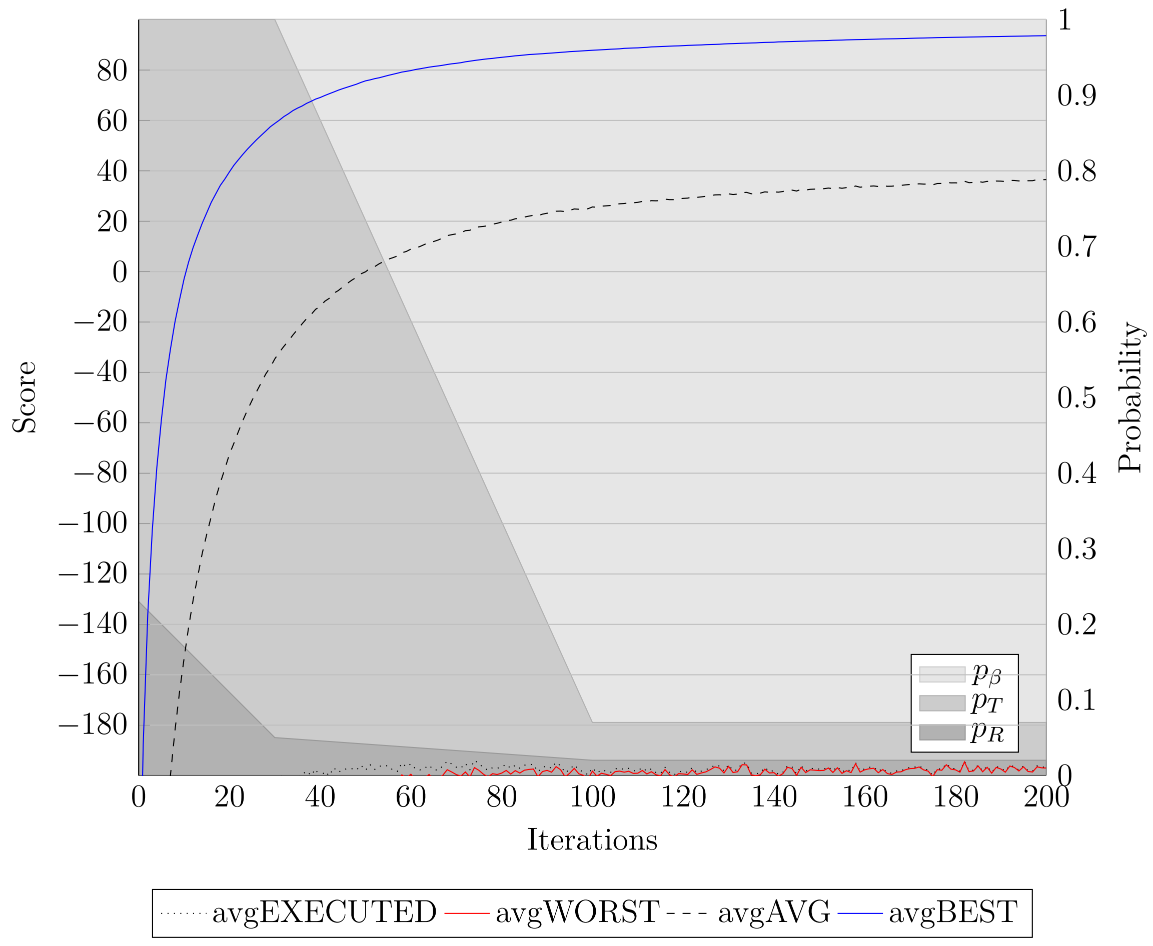
Figure 22: Congested scenario: iteration-dependent settings, first variant (Congested case)



At iteration 200, the average of the average scores is 44.65, the average of the best scores 100.27, and the average of the average scores exceeds 0 after 50 iterations.

- This configuration is composed of the optimal settings between the iteration 0 and 30, and a fraction of probability for SelectExpBeta is progressively introduced.

Figure 23: Congested scenario: iteration-dependent settings, second variant (Congested case)



At iteration 200, the average of the average scores is 36.52, the average of the best scores 93.65, and the average of the average scores exceeds 0 after 51 iterations.

The congested case does not provide much more information than on the normal scenario, which is on one hand unhelpful but on the other hand very convenient since the proposed optimal settings would not depend on the congestion of a scenario.

3.4.3 Discussion

If the settings that are commonly used in MATSim do not increase the score rapidly, how come that nobody ever changed these values? It could be the case that it belongs to a theoretical part of MATSim which has not been much investigated yet.

By simply analysing the score statistics of each simulation, the highest scores is achieved when there is no probability for the SelectExpBeta module, and it converges also faster. There are also no large oscillations when there is no execution of the SelectExpBeta, therefore this module is not necessary to reach a steady-state. The strangest observation is that if the simulation is done without SelectExpBeta and after some number of iterations the module SelectExpBeta is executed, then the average score decreases a lot. The module SelectExpBeta only selects an existing plan in the memory of an agent with a probability according to the value of the score: the higher the score the higher the probability that this plan is selected. This module reevaluates the best plan with high probability, and the value of the score is not the same after the reevaluation even if the probabilities did not change: it can be the case that an agent generates randomly a very good plan at iteration 10, since it gets a high score it will remain in the memory of the agent until it is deleted from the memory (which happens only if this plan has the worst score among all scores in the memory). Nevertheless if a plan is not reevaluated after a number of iterations, the score that this plan keeps does not correspond to the actual load of the network. In order to be up to date, the plans of the memory needs to be evaluated regularly since the other agents also change their plans at each iteration and the network load consequently changes much.

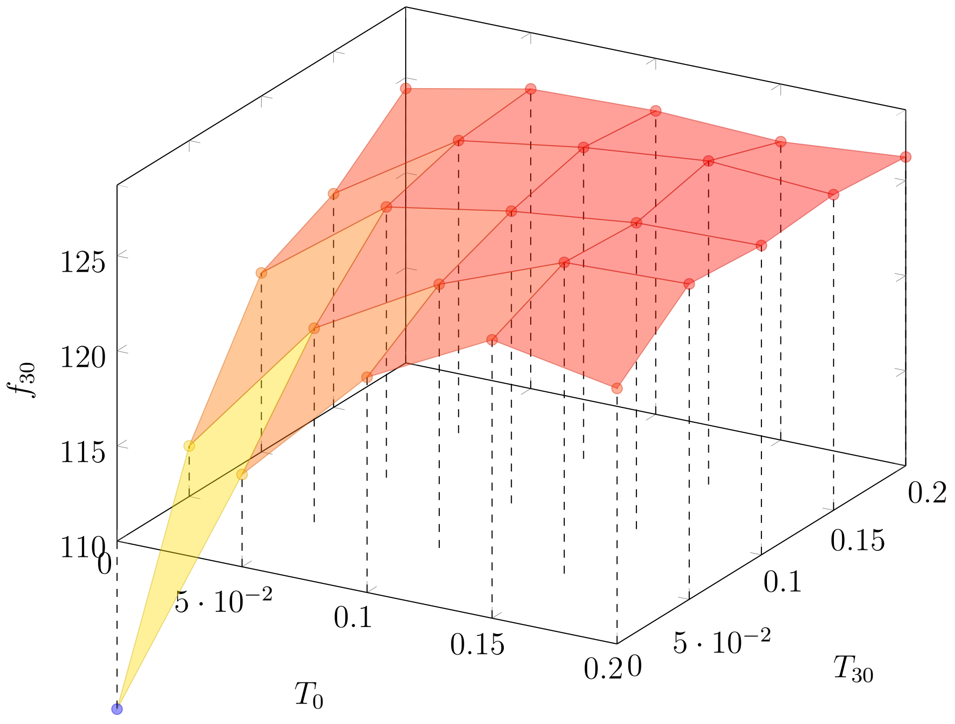
On figure 20 we see that it is once the plans are re-evaluated that the average score is shifted down. Between all iteration between 0 and 30 there was never a re-evaluation, therefore at each iteration an agent would have only generated a plan, evaluate it and then delete the plan with the worst score. But the plans in the memory will only keep the score evaluated the first time. The module SelectExpBeta aims to re-evaluate the best plans as often as possible so that the memory of an agent corresponds to the current load of the network.

Hence it is clear that without re-evaluation, each agent can produce more plans and find faster good solutions, so the iterative process will converge much faster. Nevertheless this approach creates a set of plans for each agents that will not be as good as expected since the score corresponds to another hypothetical network load (corresponding to a previous iteration). Every analytical procedure that wants to optimize the speed of convergence should consider the need of keeping the scores of each plan updated: a simulation which begins without SelectExpBeta (19) cannot re-evaluate its score at the end since it would have set aside many solutions, and running the SelectExpBeta at the end will not help to find these solutions. The average of the average of all plans in 19 is smaller than in 17 because of that problem.

There is then a need to perform a lot of re-evaluation at each iteration which is why the strategy SelectExpBeta is selected with a probability of 0.8 most of the time. An important thing to notice is that this probability should depend on the size of the memory. If an agents can keep in memory many plans there will be much more combinations with the plans of the other agents, and it will be therefore harder to have all plans updated often.

If we consider that 80% of the agents need to perform a re-evaluation of their plans at each iteration, we could search for the optimal distribution of TimeMutator and ReRouting probabilities among the 20% with the same techniques used in this project. The average of the average score at iteration 30 are plotted in 24. The higher value for the score depends on high share of the TimeMutator strategy. Moreover the score reaches its maximum 127.02 at $(T_0, T_{30}) = (0.15, 0.15)$. The commonly used settings $(T_0, T_{30}) = (0.1, 0.1)$ outputs a score of 125.37 which is smaller than the score without the ReRouting module: the average of the average score for $(T_0, T_{30}) = (0.2, 0.2)$ is 126.24. This could be explained by the fact that the

Figure 24: f_{30} for $R_0 = 0.2 - T_0$ and $R_{30} = 0.2 - T_{30}$



initial guessed routes are already very good and there is not a large need in re-routing. In order to increase the speed of convergence, the share of the ReRouting module should not exceed 10% across all iterations. Since it is also the strategy module (in the used strategies of this project) that requires the longest computation time, a share of less than 10% will also decrease the computation time.

It is harder to tell the optimal proportion of SelectExpBeta during each iteration since it is unknown yet how to specify mathematically the needs of re-evaluation for the algorithm. Moreover this proportion should surely depend on the size of the memory of each agent: if the memory size is only 2, each plan will be evaluated often, whereas if the memory contains 10 plans the re-evaluation of each plan by combination of the 10 plans of each other agent would create a huge set of possible combinations.

4. Conclusion

After testing different method to compute an optimum, it has been shown that a set of precise values for the probabilities of each strategy module does not exist, or is possibly hidden by the intensity of the noise due to randomness during the iterative procedure. A set of intervals for the probabilities were computed and improved the convergence speed a lot and also the maximum values for the score. Nevertheless these probabilities would create plans and routes that are overrated, that is the expected score would be much larger than the actual score if the final set of routes and plans are given in the input of MATSim. In fact during the iterations, the plans of each agents needs to be re-evaluated to match the current state of the traffic in the network. The module SelectExpBeta tends to re-evaluate with high probability the plans which have a high score in the memory, hence the best plans will almost always been selected and updated.

In order to improve the convergence speed without losing the consistency of the plans with the network, other strategies could be employed: a pseudo-intelligent strategy could compare the results obtained by the previously used strategies and decide which strategy to apply during the next round. The TimeMutator module is shifting the time in the schedules by a random number taken from a uniform distribution in the interval $[-30; +30]$ minutes, but perhaps after many iterations the schedule would need only refinement and no large changes. Therefore the random number for the shift could be taken from another probability distribution like a Gaussian Distribution whose variance could decrease when the iteration number increase.

Acknowledgements

I would like to thank Prof. Dr. Kay W. Axhausen for having provided a very interesting subject for this project, and yet at the same time for having given me enough freedom to let me develop and investigate solutions of my own.

I also thank my supervisor Rashid A. Waraich for all the brainstorming and discussion time which have helped a lot understanding the project and especially MATSim.

References

[Apt10] Elias Aptus. Beschleunigung der equilibriumsuche in agentenbasierten systemen. Master's thesis, ETHZ, 2010.

[Bal07] Michael Balmer. *Travel demand modeling for multi-agent transport simulation: Algorithms and Systems*. PhD thesis, ETHZ, 2007.

[GG76] K. A. Gomez and Arturo A. Gomez. *Statistical Procedures for Agricultural Research*. Wiley, 1976.

[JSW98] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *J. Global Optim.*, 13(4):455–492, 1998. Workshop on Global Optimization (Trier, 1997).

[Mei10] Konrad Meister. *Contribution do agent-based demand optimization in a multi-agent traffic simulation*. PhD thesis, ETHZ, 2010.