Exploring effects of parameter configurations on runtime, using mathematical solvers

Stefan Eggenschwiler, Michael Barry, and René Schumann

Smart Infrastructure Laboratory, Institute of Information Systems
University of Applied Sciences Western Switzerland (HES-SO) Valais/Wallis
Rue de Technopole 3, 3960 Sierre, Swizerland
stefan.eggenschwiler|michael.barry|rene.schumann@hevs.ch

Abstract. Mathematical solvers can be parameterized today with a multitude of different parameters. While default parameter settings often provide *good* results, in terms of low runtime, often parameter settings can be found, which speed-up the solving process for a particular model. Before considering the construction of strategies for optimizing parameter settings for particular models, it is necessary to understand the underlying search space. We do so by investigating systematically the effects of different parameter settings, taking into account the parameters considered to me most important in the literature. Based on three pre-existing mathematical models, we explore runtime for solving them, systematically varying the parameters of the solver. As a result of our study, we can provide a better understanding of the underlying search space, that needs to be investigated for effectively perform parameter tuning of mathematical solvers. Also we highlight that choosing *bad* parameters can have significant disadvantages, e.g. compared to the default parameters.

1 Introduction

Mathematical solvers, such as CPLEX [10], CONOPT [5], or GUROBI [6], are used to solve mathematical models in varying disciplines. The required runtime to solve a given model is largely dependent on the complexity of the model, but also on the solver's parameterization. While the *default configuration* in general provides a good performance, there often exist solver settings that can improve the runtime. Thus, finding solver parameter settings that can improve the run-time of a particular model can be considered as a search problem. Due to the observation that default parameter settings are often superior than any randomly chosen parameter setting, it raises the issue on the structure of the underlying search space. The scope of this paper is to systematically analyze the effects of different parameter settings on the runtime of solving process, and thus gives a better understanding of the underlying search space.

In the next section we discuss the currently existing approaches for minimizing the time required for solving mathematical models. In Section 3 we describe the method and experimentation set-up, used to analyze the search space for

parameter tuning. Our results are presented and discussed in Section 5. Afterwards we conclude our paper, and provide an outlook of further research. This paper is based on the BSc thesis [14] of the first author.

2 State of the art

In the following, we restrict our analysis to the use of the CPLEX solver, which is a widely used one. The process of solving a model in CPLEX can be highly customized by changing a set of parameters used at the start. The process of searching for parameterization to reduce computation time for solving a particular model, or a set of models, is referred to parameter tuning. In the following we highlight the current state of the practice as it is recommended and common upon modelers.

2.1 Recommendations for the parameter tuning

The reason to deviate from default parameter settings used by mathematical solvers is the fact that parameters have a high influence on the computation time of models. This influence can be both positive or negative, depending on how well the parameter values were chosen. Here we discuss the recommendations approaches were suggested by IBM [9].

Automated tuning tool for CPLEX The default parameter settings for CPLEX often lead to satisfactory results. But not all models run with a good-enough performance using the default settings. Since version 11.0 of CPLEX, the solver contains an automatic tuning tool. This tool will perform automated tuning tests using different parameter settings.

Node log tuning Assuming the automated tuning tool does not lead to the desired performance improvements, IBM recommends to evaluate the CPLEX node log [12]. By changing the settings of two parameters, the user can alter the log level and the log interval of the CPLEX node log. The log contains information about the progress in optimizing the original problem. By interpreting the log, it is possible to identify the cause of slow performance. This knowledge can be used to tune specific parameters in order to achieve better performance.

Probing for binary models If the model consists of binary variables, it is possible to either achieve a significant performance increase or decrease by changing the probing level of the solver. Probing is a technique used, for instance to solve Mixed Integer Linear Programming (MILP) models.

Probing is a technique performed between the pre-processing of a model and the solving of the root relaxation. It investigates the logical implications of fixing binary variables to either 0 or 1 and exploring the consequences. The intent of probing is to derive globally valid information. This can lead to a major performance increase if used on large, complex binary models. If used on

small or simple models, the probing process takes too long, thus increasing the overall computation time. In CPLEX the user can change the aggressiveness of the probing process by changing a parameter and increase the probing level. Additionally, since CPLEX version 10.0 it is also possible to set a time limit for the probing process. Even if the probing process did not finish, the information gained during the process can reduce the computing time.

Reducing the accuracy Depending on the model and its purpose, it is sometimes not necessary to find an optimal solution. By changing the emphasis of the solver it is possible to set specific trade-offs between speed, feasibility, optimality and moving bounds. This trade-off can be further refined by setting an additional condition in form of a parameter. The parameter value specifies the percentage of optimality a found solution has at least to fulfill in order to be accepted as an optimal solution.

2.2 Parameter tuning in the scientific literature

In the broader sense, also work from the field of algorithm selection, and algorithm configuration can be considered as relevant work. However, we limit us here on the specific of tuning mathematical solvers.

Hutter et al. [7] have applied Machine Learning to solver parameter tuning. Methods like [7, 15] randomly select model and solver configurations to execute and use them as training data. Furthermore, methods exist that use a multi objective approach [4], allowing a user to tune for Time-To-Optimality (runtime), Proven-Gap or Best-Integer-Solution. López and Stützle [13] also make use of an aggregate objective to find the best compromises between solution quality and runtime, to achieve good anytime behavior.

A recent review about the scientific literature, and novel approaches you can also find in [1]. Beside our study, also Baz et al. [4] showed there is great potential for using non-default settings for mathematical solvers.

3 Methodology

In the following we outline the approach we made, to investigate the overall search space, and it characteristics. In order to conduct the analysis, a key metric in the form of CPLEX Ticks has been identified. This metric provides a possibility to compare the performance of different solving runs. CPLEX Ticks are a deterministic time measurement unit introduced in version 12.4 of the solver [11]. The goal of ticks is to provide a way to compare multiple runs of the same model based on their computation effort. One drawback of CPLEX Ticks is their platform dependency which has to be taken into account during the experiment setup.

3.1 Goal question metric

To conduct an analysis, a metric has to be defined. It is used to compare the results gathered during the analysis. Goal Question Metric (GQM) is a measurement model introduced by Basili et al. [2]. A typical GQM model consists of three parts:

- **Goal:** It clarifies the basic purpose of the metric, the current issue, the object and from what viewpoint this object has to be achieved.
- **Question:** It further characterizes the object set in the goal.
- **Metric:** Defines with what means progress will be measured.

By applying the GQM approach to this topic, a GQM model can be generated for use during the analytical process. The *goal* of the analysis is to investigate the computation time of MILPs in CPLEX by deviating from the standard parameters provided by IBM. The default parameter setting will be used as a baseline for the evaluation. CPLEX Ticks are used as *metric* to compare the computational time of the models. The *question* that has to be answered is deduced from the initial goal of the analysis: How does the computation time change when custom parameter settings are applied? The difference in computation time will be measured using a percentage quotation that indicates if a custom setting is faster or slower compared to the default settings. The complete GQM model can be depicted in Table 1 below.

Table 1: Goal Question Metric Model

Goal	Purpose	Explore
	Issue	computation time of
	Object	mathematical models
	(process)	by utilizing non-default parameter settings
	Viewpoint	in IBM's CPLEX solver.
Question		What is the computation time of the model using default parameter settings?
Metrics		CPLEX Ticks
Question		How does the computation time changes using custom parameter settings?
Metrics		$\frac{\text{CPLEX Ticks with default parameter settings}}{\text{CPLEX Ticks with custom parameter settings}} * 100$

3.2 Selection of relevant parameters

As it is not feasible to test all possible combinations of the 73 parameters in CPLEX [8] it is necessary to reduce the list to a smaller set of parameters. In the following chapter, the selection criteria will be introduced and the resulting parameter set as well as the scope of the analysis will be elaborated.

Selection criteria In order to be considered as relevant, a parameter has to meet two criteria:

- Time constraints for the analysis.
- The impact a parameter has on the computation time of a model. This impact can highly variate between different models, depending on the model's structure and constraints.

STOP parameter sets We will consider two limited sets of parameters used by Baz et al. [3] in their research on automated parameter tuning using their Selection Tool for Optimization Parameters (STOP) algorithm. The first set of parameter as displayed in Table 2 consists of a total of 1296 parameter combinations. Its extended version, as seen in Table 3, is composed of 104'976 combinations.

Table 2: 6 CPLEX parameters used by STOP research [3]

#	Parameter Name	Test Settings	Default
1	MIP emphasis switch	[0,1,2,3]	0
2	MIP node selection strategy	[1,2,3]	1
3	MIP variable selection strategy	[0,2,3]	0
4	MIP dive strategy	[0,1,2,3]	0
5	MIP Gomory fractional cuts switch	[-1,0,1]	0
6	MIP MIR (mixed integer rounding) cut switch	[-1,0,1]	0

Table 3: 10 CPLEX parameters used by STOP research [3]

#	Parameter Name	Test Settings	Default
1	MIP emphasis switch	[0,1,2,3]	0
2	MIP node selection strategy	[1,2,3]	1
3	MIP variable selection strategy	[0,2,3]	0
4	MIP dive strategy	[0,1,2,3]	0
5	MIP Gomory fractional cuts switch	[-1,0,1]	0
6	MIP MIR (mixed integer rounding) cut switch	[-1,0,1]	0
7	MIP disjunctive cuts switch	[-1,0,1]	0
8	MIP cliques switch	[-1,0,1]	0
9	node presolve switch	[-1,0,1]	0
10	MIP probing level	[-1,0,1]	0

Scope of the analysis By reducing the CPLEX parameter set from 73 to six, respectively ten, we still would need to solve the same model thousands of times

during the analysis. Solving the models in a consecutive approach is infeasible due to time constraints, therefore a concurrent approach utilizing a computer cluster was used.

4 Experiment setup

4.1 Hadoop cluster

A Hadoop cluster consisting of 20 distributed slave nodes and one master node was used to conduct the analysis. Most of the slave nodes inside the cluster consist of workstation computers. In addition to those workstations, three server racks are part of the cluster. As mentioned above, CPLEX Ticks are a platform-dependent metric. This is crucial, since the cluster consists of nodes with heterogeneous hardware configurations. For this reason, the nodes with divergent hardware configuration were removed from the cluster during analysis.

Figure 1 displays how the analysis is conducted on the Hadoop cluster. A text file containing one parameter setting per line is used as input. This file is then split into chunks, each chunk corresponds to one line or parameter setting. The settings are then distributed by the master to the available slave. The slaves solve the model using the specified parameter setting and save the result as text document containing the used parameters and the amount of CPLEX ticks necessary for the computation process. After the Hadoop task concluded and all settings are calculated, the files are merged into one and ready for interpretation.

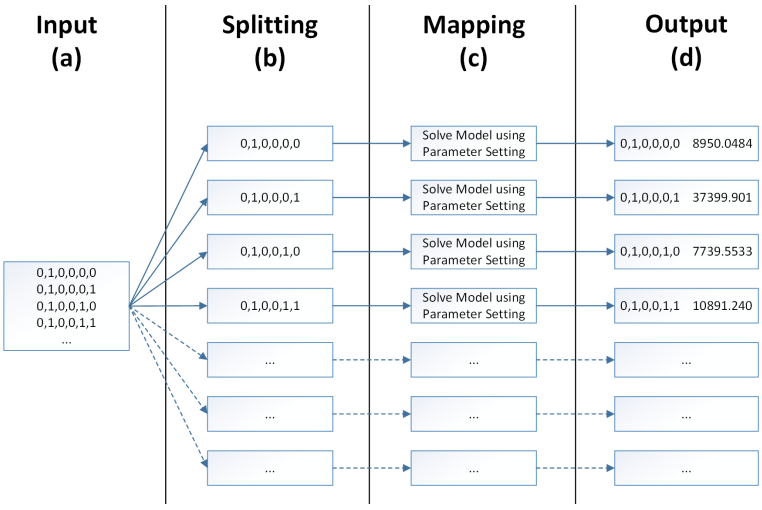


Fig. 1: Schema of CPLEX MapReduce job

4.2 Test data

The analysis was conducted using three different models that were provided by the *Industrial Process and Energy Systems Engineering (IPESE)* group at the Federal Institute of Technology Lausanne (EPFL). The models are named *Eiampl1*, *Eiampl3* and *Aladeen*. The models have different complexity with computation time using CPLEX's default settings ranging from seconds to several hours. Despite the actual purpose of those models, which is not relevant for this study, they represent rather typical models used in their research.

5 Analysis of the search space

In the following chapter, the results of the analysis will be presented and interpreted. The results of each model are represented in a box-plot with a red line marking the CPLEX ticks using default parameter settings.

5.1 Model: Eiampl1

After closer examination of Figure 2, it can be stated that over 75 percent of all computed settings yielded a worse performance than using default parameters. The computation time ranged from 19 seconds up to one hour and 17 seconds. With the default settings taking 42 seconds to compute. Only 94 out of the 1296 settings analyzed parameter settings provided an increase in performance with a maximum performance raise of up to 54.9 percent.

One possible reason why an overwhelming number of parameters did not lead to a performance increase but decrease can lie in the simplicity of the mathematical model and the default parameter values chosen by CPLEX that already provide a rather high-performance in computation time for most models.

5.2 Model: Eiampl3

The results for this mode were quite extreme as can be observed in Figure 3a. The range of results spans from 2'211 ticks up to 1.83e08 ticks. 39 out of the 1296 settings were not solved within the one-week time limit and thus do not appear in the figures.

In Figure 3b the same data is depicted with outliers hidden. Over 75 percent of all tested settings performed worse than the default settings. Only 143 out of the 1296 tested settings yielded an increase in performance of up to 27.2 percent.

5.3 Model: Aladeen

As can be examined in the box-plot in Figure 4, the distribution of CPLEX ticks for the model Aladeen is rather narrow with a small number of outliers above the upper whisker. After closer examination, we can see that the box-plot is extremely compact. The computation time ranged from 8 seconds up to one minute and 34 seconds. With the default settings taking 8 seconds to compute. 432 of 1296 results had an increase in performance with a performance increase of up to 0.22 percent.

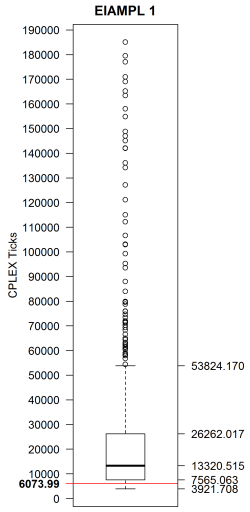
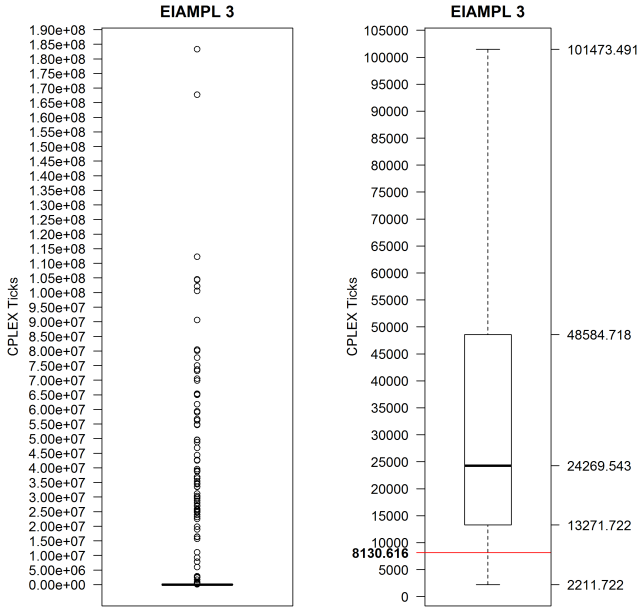


Fig. 2: Eiampl1 boxplot



(a) Full plot (b) Without outlier

Fig. 3: Eiampl3 boxplot

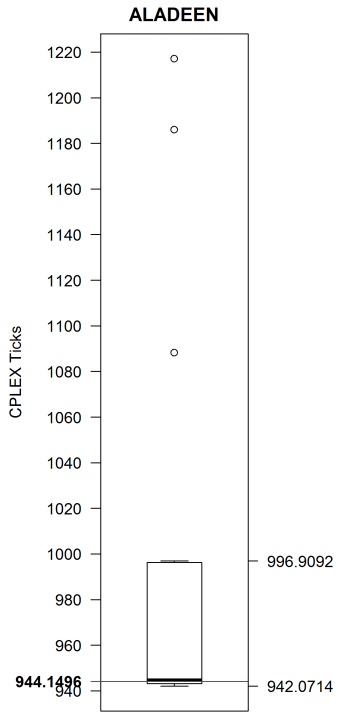


Fig. 4: Aladeen boxplot

5.4 Comparisons

In our analysis so far, we have outlined that for all models performance improvements are possible,by deviating from the standard parameters. However, we have also outlined, that most combinations lead to larger run-times, and considering the range of outliers deviations can be very time consuming.

In the following we want to investigate the structure of the search space in more detail. To allow for a more systematic comparison, we abstract now from the specific number of ticks, by just considering the factor distance from the default parameters, i.e. the factor 1 is equal to the number of ticks with the default setting, while a larger number indicates a larger runtime, while a value smaller indicates a faster execution.

We have performed this analysis for all three models, and results are shown in figures5, 6, and 7. We show a comparison between the set of parameters using the metric specified in Chapter 3.1. The different parameter settings (shown on the x-axis) have been numbered based on their sequence in the enumeration. While the upper figures show each time the full situation. i.e. runtime for the different parameter settings. The lower figures indicates for the different models evaluated a more detailed perspective, i.e. we zoomed in the areas where the performance

of parameters is better than the default parameters. Due to the rather regular patterns in the lower figures, we can conclude that there exists patterns in the parameter settings that systematically can lead to faster execution times for specific models. This can be observed over all three models in Figures 5b, 6b, and 7b.

However, we have to state that these patterns are specific for each model, as after a closer examination of the results of all three analyses, no parameter settings were found that yielded a performance improvement in all three models. Most likely, we consider that model specific aspects lead different strategy adjustments within the solver lead to an improved performance.

The overall analysis of the search space of different parameters and their effects on the runtime of the mathematical solver, we have to state that the due to the quality of the default parameters, and the often quite big risk of misguessing, the underlying search space is highly imbalanced, i.e. the *interesting* parts of the search space, where the runtime is lower than the default parameters is quite small and typically centered around specific key parameters that are most relevant for a particular model, that finding them can be considered difficult, and also optimization and learning strategies should take into account the imbalanced nature of the underlying search space.

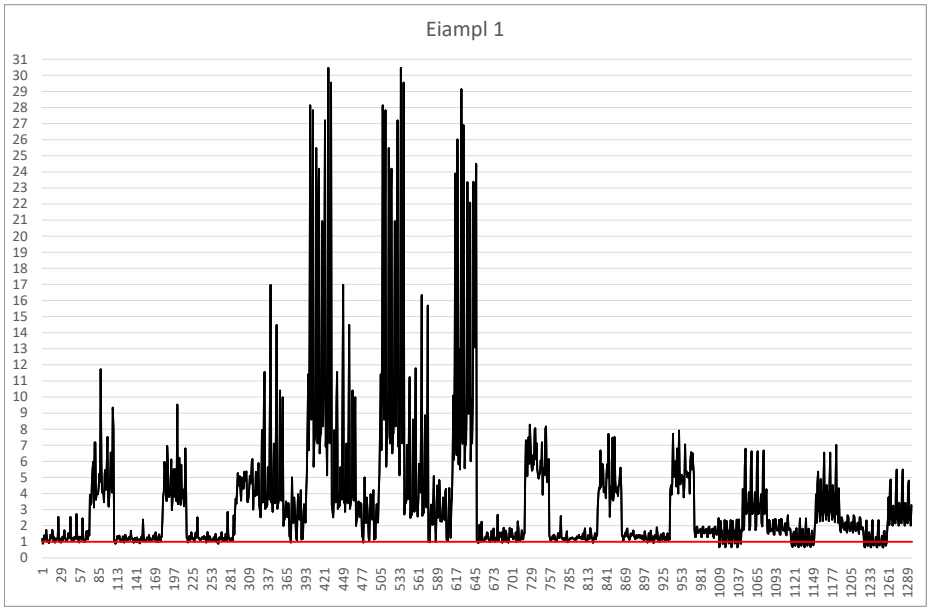
6 Conclusion

Due to a large number of configurations that are possible and the fact that most solver parameter settings will result in an exceptional high runtime, the sampling space is large and imbalanced. Further strategies that consider optimizing parameter settings have to take into account the structure of the underlying search space. We have seen for all models that we very likely can find improvements, along some patterns of parameter settings. However, as there exists no one-fits-all solution, we have to conclude that parameter adjustment must be done at least specific for the model at hand. This challenge is still large, due to the imbalanced structure of the search space, i.e. desired parameter settings are rare, and a potential high penalty (runtime) if parameters are mis-guessed.

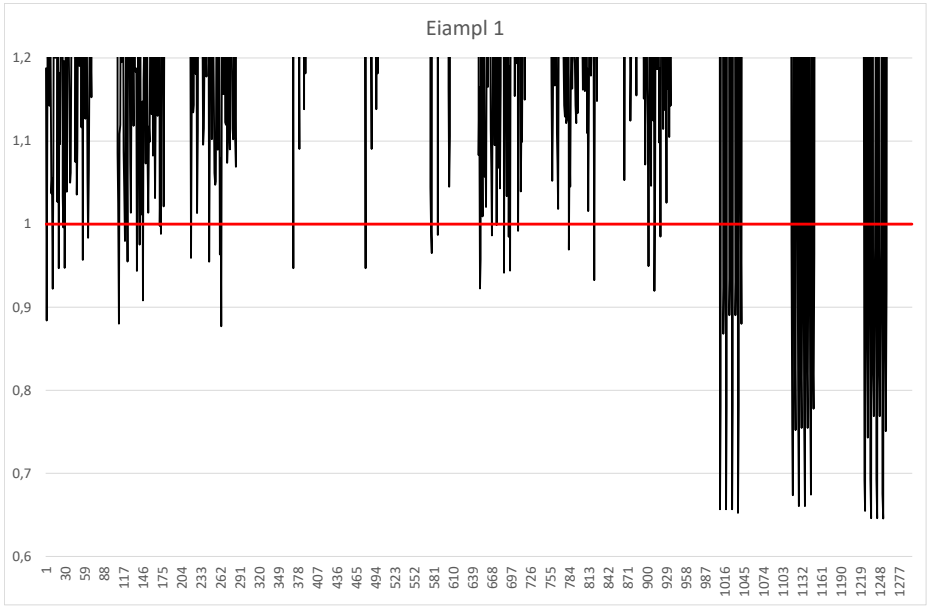
In our future research we are considering mechanisms how parameter-tuning can be obtained, thereby considering the underlying search space.

References

1.
2. Basili, V.R., Caldiera, G., Rombach, D.H.: Goal Question Metric Paradigm. John Wiley & Sons, Inc. (1994)
3. Baz, M., Hunsaker, B., Brooks, J.P., Gosavi, A.: Automated Tuning of Optimization Software Parameters. Tech. rep. (2007)
4. Baz, M., Hunsaker, B., Prokopyev, O.: How much do we "pay" for using default parameters? Computational Optimization and Applications **48**(1), 91–108 (2011)
5. Drud, A.: Conopt solver manual. ARKI Consulting and Development, Bagsvaerd, Denmark (1996)

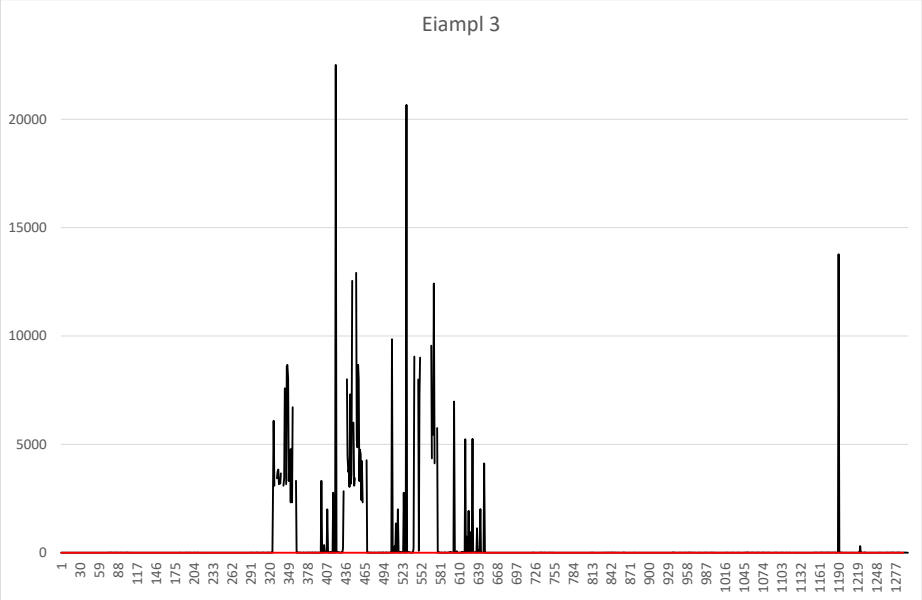


(a) Complete solution set

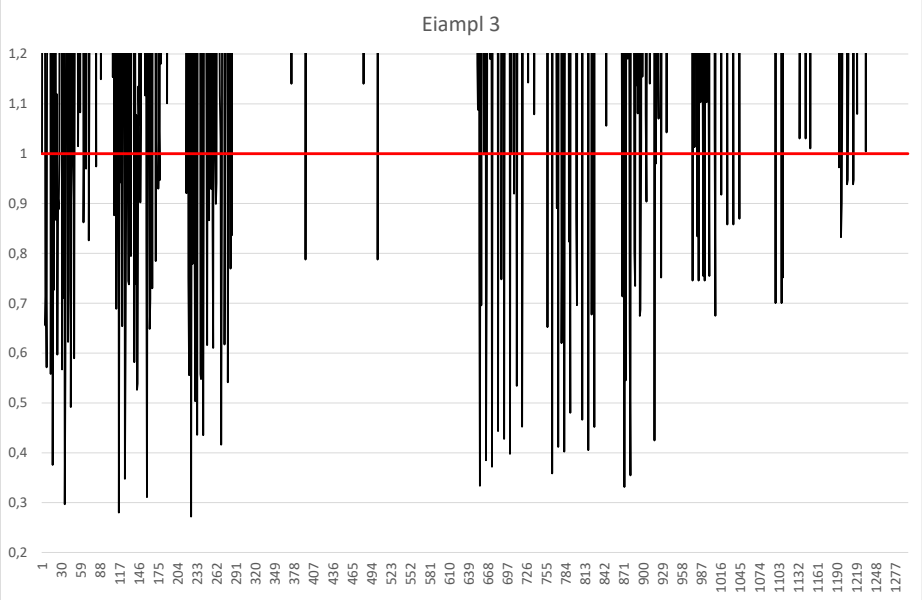


(b) Positive solution set

Fig. 5: Eiampl1 GQM comparison

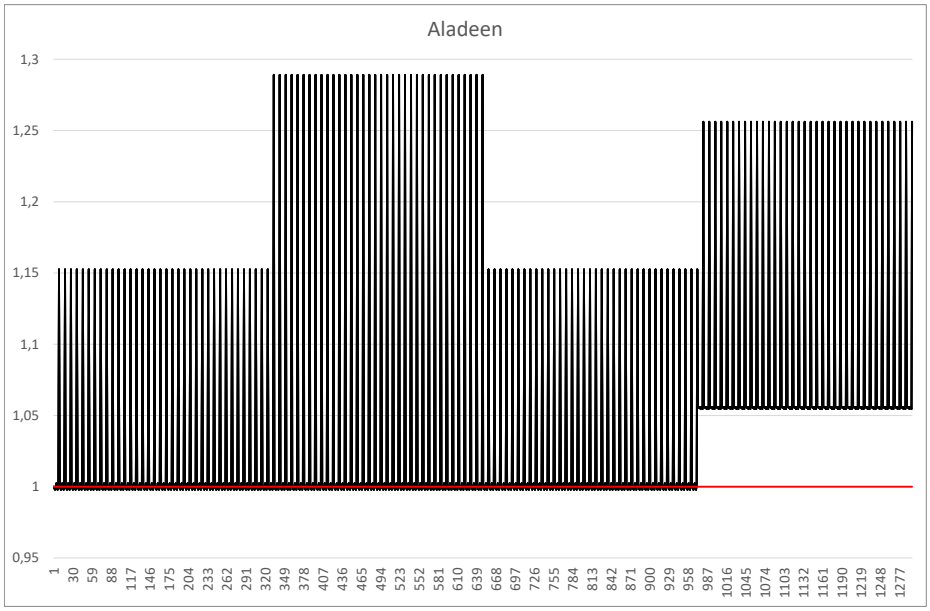


(a) Complete solution set

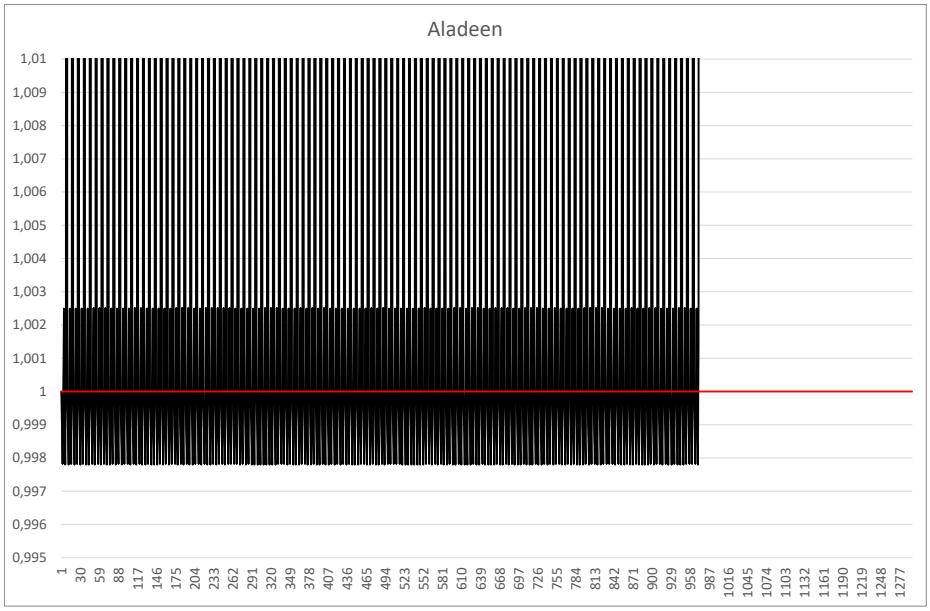


(b) Positive solution set

Fig. 6: Eiampl3 GQM comparison



(a) Complete solution set



(b) Positive solution set

Fig. 7: Aladeen GQM comparison

6. Gurobi Optimization, I.: Gurobi optimizer reference manual (2016), http://www.gurobi.com
7. Hutter, F., Hoos, H., Leyton-Brown, K.: Automated configuration of mixed integer programming solvers. Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems p. 190 (2010)
8. IBM: CPLEX Parameters Reference, Version 12 Release 6, http://www.ibm.com/support/knowledgecenter/SSSA5P_12.6.2/ilog.odms.studio.help/pdf/paramcplex.pdf
9. IBM: IBM CPLEX Performance Tuning for Mixed Integer Programs, https://www-01.ibm.com/support/docview.wss?uid=swg21400023
10. IBM: IBM ILOG CPLEX Optimization Studio CPLEX User's Manual https://www.ibm.com/support/knowledgecenter/SSSA5P_12.6.1/ilog.odms.studio.help/pdf/usrcplex.pdf
11. IBM: IBM Knowledge Center - deterministic time limit, https://www.ibm.com/support/knowledgecenter/SS9UKU_12.5.0/com.ibm.cplex.zos.help/Parameters/topics/DetTiLim.html
12. IBM: IBM Knowledge Center - Progress reports: interpreting the node log, https://www.ibm.com/support/knowledgecenter/SSSA5P_12.5.1/ilog.odms.cplex.help/CPLEX/UsrMan/topics/discr_optim/mip/para/52_node_log.html
13. López-Ibánez, M., Stützle, T.: Automatically improving the anytime behaviour of optimisation algorithms. European Journal of Operational Research **235**(3), 569–582 (2014)
14. Stefan Eggenschwiler, R.S.: Parameter tuning for the cplex. Bachelor Thesis (2016)
15. Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Hydra-mip: Automated algorithm configuration and selection for mixed integer programming. In: RCRA workshop on experimental evaluation of algorithms for solving problems with combinatorial explosion at the international joint conference on artificial intelligence (IJCAI). pp. 16–30 (2011)