INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Calibrating Traffic Simulations as an Application of CMA-ES in Continuous Blackbox Optimization: First Results

Dimo Brockhoff, Anne Auger, Fabrice Marchal

N° 7304

June 2010

Domaine 1

ISSN 0249-6399 ISRN INRIA/RR--7304--FR+ENG

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE
INRIA
centre de recherche SACLAY - ÎLE-DE-FRANCE

Calibrating Traffic Simulations as an Application of CMA-ES in Continuous Blackbox Optimization: First Results

Dimo Brockhoff*, Anne Auger†, Fabrice Marchal‡

Domaine : Mathématiques appliquées, calcul et simulation
Équipes-Projets Thème Apprentissage et Optimisation (TAO)

Rapport de recherche n° 7304 — June 2010 — 26 pages

Abstract: Estimating car traffic is crucial in many big cities around the world to provide users with good alternatives for their travels but also to help decision makers while evaluating the impact of road pricing, new roads, or short-term road works. It will, in the best case, reduce traffic jams drastically—resulting in a significantly smaller production of atmospheric greenhouse gases. The estimation of traffic relies thereby on a precise monitoring of the current traffic as well as on a simulation which can reproduce the observed data reliably. To match the observed traffic and the simulation outputs, an optimization of parameters of both the model and the simulator itself is necessary.

Within this paper, we present a first study of how state-of-the-art evolutionary computation approaches can be employed in such a scenario. In particular, we use the well-known Covariance Matrix Adaptation Evolution Strategy (CMA-ES) to calibrate the dynamic car traffic simulator *Metropolis*. As the calculation of the objective function is expensive, a parallel evaluation of the fitness is implemented. First experiments for a simplified city network of Sioux Falls, SD, USA show that the approach is working in principle but also that the objective function contains noise. An easy way to deal with noise within CMA-ES is to simply increase the population size—experiments on easy-to-calculate noisy test functions support this impact exemplary. Additional traffic calibration runs with larger population size, however, do not support this impact but

This work receives support by the French national research agency (ANR) within the SYSCOMM project ANR-08-SYSC-017.

* INRIA Saclay—Île-de-France, projet TAO, Bat 490, Université Paris-Sud, 91405 Orsay Cedex, France, dimo.brockhoff@inria.fr, http://www.lri.fr/∼brockho/

† INRIA Saclay—Île-de-France, projet TAO, Bat 490, Université Paris-Sud, 91405 Orsay Cedex, France, anne.auger@inria.fr, http://www.lri.fr/∼auger/

‡ Orell Füssli, Wirtschaftsinformationen AG, Hagenholzstrasse 81, 8050 Zurich, Switzerland

Centre de recherche INRIA Saclay – Île-de-France
Parc Orsay Université
4, rue Jacques Monod, 91893 ORSAY Cedex
Téléphone : +33 1 72 92 59 00

make us believe that numerical precision problems within the simulator are the reason for the noise observed.

Key-words: traffic simulation, continuous optimization, CMA-ES, application

1 Introduction

Many big cities are facing problems with increasing car traffic nowadays. In this respect, it is crucial to predict the car traffic as precisely as possible to increase traveler satisfaction on the one hand but also to reduce greenhouse gases and noise emissions by reducing the amount of traffic jams when the cars can be guided according to the predictions. In addition, the influence of road pricing, additionally build roads but also short-term events such as road works needs to be investigated—not only in highly congested areas. To do so, and similar to other application areas such as weather prediction, statistical models of the traffic flow have to be established: only modeling the observed data as good as possible allows to later predict the future by simulating the model under different conditions. This approach of modeling and simulating the traffic by means of statistical models almost always includes optimization of the models' and/or the simulators' parameters. Moreover, parameter calibration within traffic simulation is a well-suited optimization scenario for evolutionary computation (EC) methods as the objective function, i.e., usually the fit between measured and simulated data, is computationally expensive and—since the computation involves a simulation—no knowledge about the objective function itself can be used to guide the search (blackbox scenario).

This paper presents, as an example, the calibration of the mesoscopic traffic simulator Metropolis [3] as a real-world application for the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [8] where the simulations are run in parallel on a grid to reduce the computation time for a single generation of CMA-ES.

In particular, we present, besides the detailed description of the optimization algorithm including the parallelized simulations (Sec. 5) and the optimization problem itself (Sec. 4), first results on the calibration of the Metropolis simulator (Sec. 6). It turns out that the calibration works in principle but also that the objective function is noisy. An easy approach to deal with noisy objective functions with the CMA-ES is to simply increase the population size as an increased population size has been shown to result in a quality increase of the found solutions already on highly multimodal functions [8]. That this has a positive effect on the obtained results is first shown on fast-to-calculate noisy test functions within the black box optimization benchmarking framework described in [7] (Sec. 7) before additional experiments on the Metropolis calibration are presented (Sec. 8). Interestingly, the experiments on the practical calibration problem of Metropolis do not show the same behavior which, together with further experiments, let us come to the conclusion that the *noisy* objective function might be mainly caused by numerical errors of the simulator. Before to start, we introduce the Metropolis simulator in greater detail (Sec. 2) and quickly review related applications of EC methods in traffic optimization (Sec. 3).

2 Mesoscopic Traffic Simulation: Metropolis

A traffic model usually consists of the two components *demand model* and *supply model* and is often described in the four stages (i) trip generation, (ii) trip distribution, (iii) modal split, and (iv) assignment [9]. Such a model describes

the traffic flow within a network implicitly by its equilibrium point where supply and demand match which can be found experimentally by simulating the model.

Depending on the accuracy of the modeled phenomena, traffic models can be typically divided into three classes: macroscopic, microscopic, and mesoscopic models. *Macroscopic models* use only macroscopic properties of the network such as flow and density to describe the congestion and do not concentrate on the individual parameters such as driver behavior etc. Instead, these parameters of the traffic are aggregated over all users in comparison to *microscopic models* where vehicles are described individually. Microscopic simulations are quite useful to model small systems such as traffic signals, junction interactions etc. If, however, an entire city network has to be modeled, the high number of streets and junctions requires expensive and time-consuming computation. A compromise between these two extremes is provided by so-called *mesoscopic models*. They have two main properties: On the one hand, the supply model uses aggregated speed-density laws instead of the detailed aspects of microscopic models and on the other hand, for the demand model individual decisions are taken into account.

Metropolis [3] is an example of a traffic simulator for dynamic car traffic based on a mesoscopic traffic model that simulates the traffic flow in a network based on agents where each agent corresponds to a single car.

As we cannot describe the full model here, we refer to [3, 2] for details. On a rough scale, the agents start their paths through the network with a stochastic departure time choice and with only a naive knowledge of the network and ignore congestion in the beginning. Then, for each choice, e.g., which road to take next, an agent uses some global information, e.g., about the traffic flows, stored separately in the model. This global information is updated after every iteration of the simulation, i.e., after each simulated day. The agents then learn from their day-to-day travel experiences and incurred traffic congestion over a period of 100 days which coincides with finding an equilibrium of the model. For each road section, a FIFO queue contains the agents at each time step of the simulation where the traffic flow is ensured to always lie below the road capacity. Jams are therefore propagated to adjacent links.

3 EC Applications in Traffic Optimization

Applications of evolutionary computation (EC) approaches to the calibration of traffic simulators are surprisingly sparse. Most of the studies that can be found in the literature optimize discrete microscopic models via genetic algorithms, in particular for the optimization of traffic light cycles [17, 11, 14]. To the best of our knowledge, no continuous calibration of traffic simulator parameters has been performed with EC methods with the exception of [16] where a genetic algorithm with a discrete representation has been used to optimize a continuous calibration problem which resulted in slightly worse results than if the algorithm SPSA [15] was used. The advantage of using the state-of-the-art stochastic algorithm CMA-ES [8] for the calibration of metropolis, when compared to classical, deterministic optimization techniques such as SPSA, is the fact that CMA-ES is able to deal with both non-separable and noisy functions whereas SPSA is not, see for example the results of the blackbox benchmarking exercise of the BBOB workshop series [6, 4, 5].

4 Optimization Scenario

Optimally, the outcome of a traffic simulator should comply with the data observed in practice. Here, we consider a slightly artificial scenario in a sense that we assume we know the actual travel times for certain times of the day on three distinct paths in the network. The idea behind this scenario is that we could get this type of information about the actual traffic flows in a real-world scenario by measuring the traveling times with GPS-equipped cars that follow the specific paths[1].

More specifically, we consider the simplified street network of Sioux Falls, SD, USA [10] and the three paths depicted in Fig. 1. The next subsections explain in more detail which parameters of the car traffic simulator Metropolis are optimized (Sec. 4.1) and how the objective function for the optimization looks like (Sec. 4.2).

4.1 Optimized Parameters

In our preliminary study, only six parameter have been optimized to keep the problem reasonably small. Usually, much more parameters are unknown in practice such as the O-D matrix, which results by itself in up to $n(n-1)$ additional parameters if n is the number of road junctions. Also the capacities of the road sections could serve as additional $\Theta(m)$ parameters if m is the number of roads. Here, the calibrated variables are

- the total demand N over the peak period, i.e., the amount of cars in the simulation,
- a multiplicative factor K indicating the flow capacity of the roads such that each road in the network has the same capacity,
- the average vehicle length L which determines how many cars fit in a certain road section without traffic jam,
- the starting time T_0 of the simulation,
- the ratio R of peak and off-peak travel, i.e., the percentage of home-to-work commuting trips among all moving cars, and
- a scaling parameter μ which influences the single car's choice of their departure times.

Note that, although six parameters are exposed to the optimization algorithm, there are only four independent variables: in the current implementation, the vehicle length is not influencing the objective function to see how the optimization algorithm is dealing with this and the parameters N and K are highly correlated[2] to investigate whether the optimization algorithm can detect that.

[1] Note that, in principle, any other real-world data, such as traffic flow measurements on single road segments, could be integrated into the calibration approach as well.

[2] Doubling, for example, the number N of vehicles and at the same time doubling the flow capacity K will result in (almost) the same simulation output regarding the travel times. Only if the number of cars and the flow capacity are small, the random choices of the agents in the simulation will have a significant influence on the observed travel times. As we have about $70,000$ cars in our simulations, we expect the two parameters to be almost fully correlated.

To be more precise, the six real-valued variables x_N, x_K, x_L, x_T, x_R, and x_μ corresponding to the above described parameters and used within the optimization algorithm are all randomly chosen within the interval $[0, 1]$. For the simulation, these parameters (except for the fifth parameter R which itself is a ratio between 0 and 1) are then transformed for the simulations to the correct value in the following ways.

Total demand N The basic amount of cars is given by the O-D matrix which is kept fixed over time and which corresponds to about $70,000$ cars. The variable x_N is rescaled to the factor $y_N = 1 + \frac{(60 * x_N) - 30}{100}$ by which the amount of cars given by the O-D matrix is multiplied to yield N.

Flow capacity K The flow capacity is kept the same for each road section and the variable x_K is rescaled to the factor $y_K = 1 + \frac{(60 * x_K) - 30}{100}$ by which this baseline capacity is multiplied to yield the actual value K for the simulation.

Vehicle length L The vehicle length L (in m) is obtained from the variable x_L as $L = (18 * x_L) + 2$.

Starting time T Based on a base scenario starting at 8 o'clock, the variable x_T is used to compute the offset time T_{off} (in min) as $T_{\text{off}} = (120 * x_T) - 60$.

Scaling parameter μ The scaling parameter μ is obtained by $\mu = (4.8 * x_\mu) + 0.2$

Moreover, the integrated boundary handling of CMA-ES ensures that the parameters are within the interval $[0, 1]$.

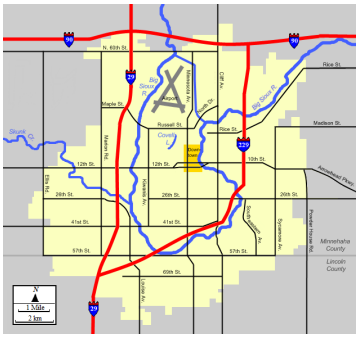
4.2 Objective Function

The main idea behind the objective function is to compare the simulated travel times on each road segment with the observed data from practice. It is expected that the data cannot be provided for each single road section as it is too expensive to install the measuring devices on every road. Instead, we assume that GPS-equipped cars travel through the network and tell us individual travel times for selected car trips. As the real-world data is not available at the moment, we make a simplification here and focus on three distinct but fixed paths within the network at the moment. For the road sections along those three paths, we fix the desired travel times and define the objective function as the square root of the sum of the squared errors between simulated travel times and the target travel times on these paths[3].

The underlying network is a simplified version of the street network within the city of Sioux Falls, SD, USA [10]. Figure 1 shows both the network itself as well as the selected three paths for the objective function.

The base case scenario (BC) consists of a morning peak period with about $70,000$ individual simulated car trips. A hundred iterations of the day-to-day

[3]The desired travel times have been obtained by an independent simulation with parameters close to the corner of the search space: $x_N = ((1.2228 - 1) \cdot 100 + 30)/60 \approx 0.87133$, $x_K = ((0.8 - 1) \cdot 100 + 30)/60 \approx 0.16667$, $x_R = 0.3845$, $x_T = 105/120 = 0.875$, $x_\mu = 3.8/4.8 \approx 0.79167$. The paths have been selected by hand.



© Jon Platek, taken from
http://commons.wikimedia.org/wiki/
File:Sioux_Falls_Map_4.png

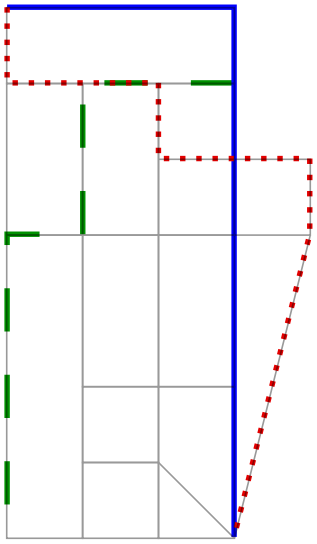


Figure 1: Street map of Sioux Falls, SD, USA (left) and corresponding simplified road network with the three paths (dashed, dotted, and bold) used within the objective function (right).

adjustment process are run in order to insure proper convergence of the simulation. In principle, the simulator results are stochastic and therefore noisy by default. Here, however, the random seed for the pseudo-random number generator has been fixed artificially which somehow "freezes" the noise to a deterministic (but at the same time rugged) objective function.

5 Running the CMA-ES With Parallelized Function Evaluations

5.1 The Basic CMA-ES

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is a stochastic optimization algorithm where at each iteration g, a population of λ points is sampled according to a multivariate normal distribution [8]. The objective function of the λ points is then evaluated and the parameters of the multivariate normal distribution are updated using the feedback obtained on the objective function. More specifically, let $(\mathbf{m}_g)_{g\in\mathbb{N}}$ be the sequence of mean values of the multivariate normal distribution generated by CMA-ES, constituting the sequence of estimates of the optimum and let $(\sigma_g)_{g\in\mathbb{N}}$, and $(\mathbf{C}_g)_{g\in\mathbb{N}}$ respectively, be the sequences of step sizes and covariance matrices. Given the values $\mathbf{m}_g$, σ_g, and $\mathbf{C}_g$ for generation g, new points or *individuals* are sampled according to

$$\vec{x}_i^{\,g} = \mathbf{m}_g + \sigma_g \mathcal{N}_i(0, \mathbf{C}_g), \qquad \text{for } i = 1 \ldots \lambda \ , \tag{1}$$

where $(\mathcal{N}_i(0, \mathbf{C}_g))_{1\leq i\leq\lambda}$ are λ independent multivariate normal distributions with zero mean vector and covariance matrix $\mathbf{C}_g$. Those λ individuals are

ranked according to f:

$$f(\vec{x}^{\,g}_{1:\lambda}) \leq \ldots f(\vec{x}^{\,g}_{\mu:\lambda}) \leq \ldots f(\vec{x}^{\,g}_{\lambda:\lambda}) \tag{2}$$

where we use the notation $\mathbf{x}^g_{i:\lambda}$ for the i^{th} best individual. Moreover, we denote the best individual $\mathbf{x}^g_{1:\lambda}$ in generation g as $\vec{x}_{\text{best},g}$ and the individual $\mathbf{m}_g$ also as $\vec{x}_{\text{mean},g}$. We omit the generation index for clarity whenever it is clear from the context.

During the algorithm, $\mathbf{m}_g$, σ_g, and $\mathbf{C}_g$ are updated in a deterministic manner according to the objective values of the visited points. We refer to [8] for the equation updates. All updates rely on the ranking determined by Eq. 2 only and not on the exact value of the objective functions such that the CMA-ES is invariant when optimizing f or $g \circ f$ where $g : \mathbb{R} \mapsto \mathbb{R}$ is a strictly increasing mapping. The default population size λ equals $4 + \lfloor 3\ln(n) \rfloor$ and is fixed during the run. To shorten the computation time per generation, a parallelized version of CMA-ES has been used as explained in the following.

5.2 Distributing Expensive Function Evaluations on a Grid: The BBOG Framework

The Blackbox Optimization Grid, or BBOG for short, is a grid service for optimizing expensive objective functions. The framework of BBOG decouples algorithms and problems and as such allows for easy benchmarking. Written in Java, the BBOG framework is focused on single-objective optimization and does not do the communication via text files as for example the PISA framework [1] but via a MySQL database instead. The main focus of BBOG, however, is not only the easy benchmarking, which nowadays can be done with less implementation effort for example with the Blackbox Optimization Benchmarking (BBOB) framework [7] but at the distribution of expensive objective function evaluations on a grid. Other frameworks with similar objective can be found in the literature. However, they either do not provide a CMA-ES implementation (such as ParadisEO-CMW [12]) and/or are not easily compatible with the Java based Metropolis simulator (such as ParadisEO-CMW, written in C++, or pCMALib [13], written in Fortran).

The main difference between the above described unparallelized CMA-ES version of [8] and the one implemented in the BBOG framework is the way the objective function values are obtained. In our specific implementation, only the Metropolis simulations themselves are run in parallel on so-called *clients* whereas the computation of the objective function value itself as well as the algorithm are not seen as time critical and are therefore run on a dedicated machine (the *master*). In more detail, the new individuals of CMA-ES are generated and the corresponding parameters are computed as described in Sec. 4.1 on the master sequentially. Then, the grid service of BBOG, also running on the master, generates a list of λ Metropolis simulations ready for running on the grid and writes them into a central MySQL database. Depending on the available computing resources, the first N simulations in this list are then send to the N free processors of the clients and run there in parallel. Once a simulation is finished, the results are written in the MySQL database and BBOG ensures that CMA-ES is notified in order to evaluate the objective function based on the obtained travel times which takes place again on the master. After all λ

simulations are completed and the objective function values are computed, the CMA-ES is continued as usual with the next iteration.

In our particular case, the grid contained (at most[4]) the following processors:

- 2 64bit AMD Opteron 250 processors with 1.8GHz
- 10 64bit AMD Opteron processors with 2.4GHz
- 16 64bit AMD Opteron processors with 2.5GHz and 4-cores each.

The additional master had a 64Bit AMD Opteron with 2.33GHz. A single Metropolis simulation, i.e., a single objective function evaluation, took on average about 10 minutes.

6 First Experimental Result

The experiments presented in the following serve several purposes. On the one hand, we would like to learn about the problem in a first step; on the other hand, the experiments should show that calibration is in principle possible. In particular, we would like to check whether CMA-ES can identify that (i) the vehicle length L is not influencing the objective function and that (ii) the two parameters total demand N and flow capacity K are highly correlated.

To this end, a run of CMA-ES with a fixed population size of $\lambda = 24$ has been run where the number of 24 samples in each generation is due to the number of available processors in the grid at the time of the start. The parent population size has been always chosen as $\lambda/2$ throughout the paper as recommended. Other parameters of the CMA-ES in this and the following runs which are different to the standard settings of [8] are an initial step size of $\sigma = 0.3$, stopFit $= 10^{-6}$ and StopTolFun $= 1^{-12}$.

Figure 2 shows the typical output of the CMA-ES algorithm: Most interesting are the evolution of the current best objective function value over the number of function evaluations (upper left plot, in blue) and the six (unscaled) parameters over time (upper right plot). The two plots at the bottom show the search distribution. In particular, we can see the square root of the eigenvalues of the covariance matrix in the lower left plot and the square root of its diagonal entries in the lower right plot.

The main observations of Fig. 2 are: (i) the objective function could be improved (blue line in upper left plot), although we see (ii) a stagnation together with a kind of noise or in other words a high variance after the first approximately 1000 function evaluations, and moreover (iii) the parameter values (upper right plot) show a convergence towards a fixed value except for the independent parameter x_L—indicating that the CMA-ES found a (local) minimum of the objective function. To further investigate the parameter values found, Fig. 3 shows the evaluation of the (recombined) parameter value $\vec{x}_{\text{mean}}$ within CMA-ES over time together with the optimal parameter value implanted in the objective function. In accordance with the upper left plot of Fig. 2 where the objective function value drops fast within the first 1000 function evaluations, we see in Fig. 3 that the optimization also quickly converges to the optimal parameter value implanted in the objective function. A further improvement of

[4]Not all machines have been available during the entire time the experiments have been conducted (December 2009 and January 2010).

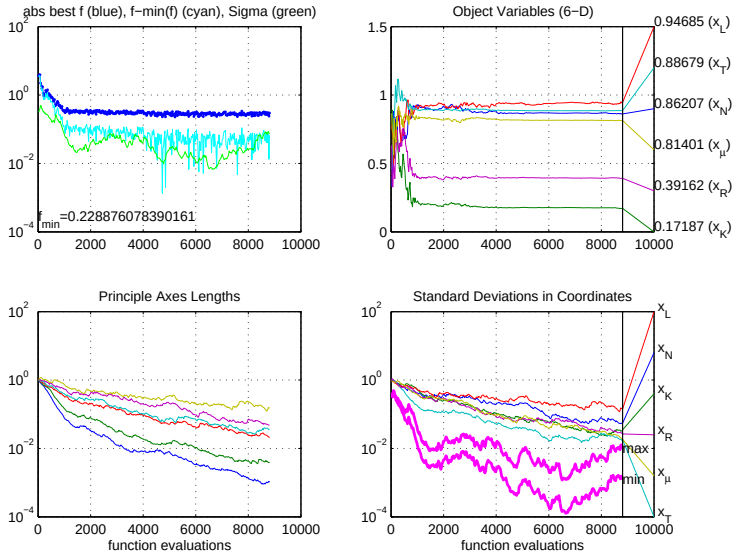


Figure 2: Standard output of the CMA-ES for the first run with a population size of $\lambda = 24$. Most interesting are the evolution of the current best objective function value over the number of function evaluations (upper left plot, in blue) and the six (unscaled) parameters over time (upper right plot). The search distribution can be seen in the lower two plots: square root of the eigenvalues of the covariance matrix (lower left) and the square root of its diagonal entries (lower right).

the objective function after the first about 1000 fitness evaluations, however, seems not to be possible due to the high variance in the objective function—an observation which one would usually call a *noisy objective function*. Our goal in the remainder of the paper is to investigate this "noise" further to understand the underlying properties of the calibration problem and to be probably able to improve the convergence of CMA-ES to the optimal solution further.

7 Investigating the Influence of CMA's Population Size on the BBOB Testbed

As we have seen in the previous section, the simulations add a kind of noise to the objective function evaluation which at first sight could be caused by the random decisions of the agents within the simulation. However, the random number generator within the simulator is always instantiated with the same seed such that we do not have a noisy objective function in the strong sense: if we evaluate a parameter set several times, we will always get the same objective function value. Instead, either the granularity, i.e., the discreteness, of the simulation adds a kind of noise or random numerical errors result in what we see as noise[5].

[5]This influence of numerical errors on the objective function can be also observed on test functions for which the optimum does not lie in 0, e.g., the Rosenbrock function.

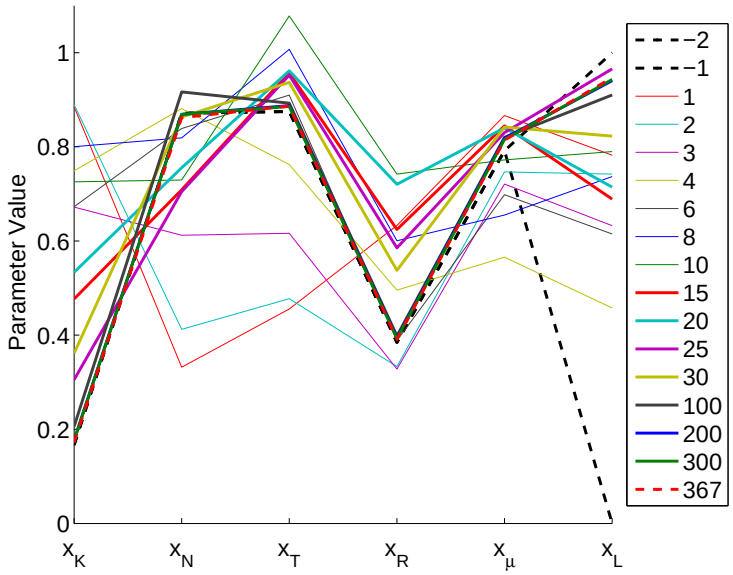


Figure 3: Evolution of the parameter value $\vec{x}_{\text{mean}}$ over time for the calibration run with population size $\lambda = 24$. The parameter values are only plotted for some generations, given by the numbers in the legend where -1 and -2 correspond to the optimal parameter value encoded in the objective function with maximal and minimal possible values in the parameter x_L that has no influence on the objective function.).

Although we do not know which type of noise we see at the moment, the first idea is to deal with it by increasing the population size of CMA-ES—a simple way to increase the quality of the found solutions on highly multimodal functions [8]. To investigate this increase in population size, we run the standard CMA-ES with fixed population size of [8] with different population sizes on the noisy black box optimization benchmarking framework BBOB—consisting of 30 test functions with different noise types in dimensions 2, 3, 5, 10, 20, and 40 [7]. Three different variants of the CMA-ES have been tested: the one with standard population size, i.e., $\lambda_s = 4 + \lfloor 3 \ln(\text{dimension}) \rfloor$, a version with population size $16\lambda_s$, and a version with an unusually high population size of $128\lambda_s$. For each of the 5 instances of each function, maximally $10^4 \cdot \text{dimension}$ function evaluations have been allowed. The results can be seen in Fig. 4 to 9.

When looking carefully at the results, we see two main impacts of the increase in population size: on the one hand, more functions can be solved with a predefined precision within the given budget and, on the other hand, the amount of function evaluations needed to solve the problems gets lower with increasing population size for some functions.

8 Calibration Results With Larger Population Size

With the observation that increasing the population size is beneficial on noisy test problems as we have seen above, the question arises whether a larger population size can also improve the results on the metropolis calibration problem. To this end, we started several calibration runs with increasing population sizes as detailed in Table 1—with population sizes up to 200 and up to 25200 function evaluations per run[6].

Figures 11 and 12 show exemplary the output of CMA-ES for the runs with population size $\lambda = 48$ and $\lambda = 200$. Also here, we can see that CMA-ES is optimizing the objective function but the amount of variance in the objective values over time does not seem to be reduced in comparison to Fig. 2. In terms of the parameter values found by CMA-ES, we can observe for all runs a tendency towards the optimal values, i.e., the parameter values implanted into the objective function which theoretically should give an objective function value of 0—keeping in mind that the parameter values are chosen uniformly at random in the beginning. Figure 13 shows the parallel coordinates plots of both the parameter values yielding the optimal objective function value in each run ($\vec{x}_{\text{best}}$) and the parameter values corresponding to the recombined search point $\vec{x}_{\text{mean}}$ in the last generation of the algorithm. Coming back to the initial purpose of our experiments, stated in the beginning of Sec. 6, we see that the CMA-ES can find both (i) that the parameter x_L has no influence on the objective function (different values are found in every new run) and (ii) that the parameters x_N and x_K are highly correlated (the ratio of them is nearly the same in all found solutions depicted in Fig. 13). Note that for a solution $\vec{x}_{\text{mean}}$,

[6] As the evaluation of the objective function is expensive, only a few runs have been done and some of the runs have been stopped by hand as soon as no improvement could be seen in the output of the optimizer CMA-ES. The run with $\lambda = 24$, e.g., took about 27 hours of parallel computation for the $368 \cdot 24 = 8832$ evaluations—resulting in an average computation time of approx. 4.4 minutes per evaluation.

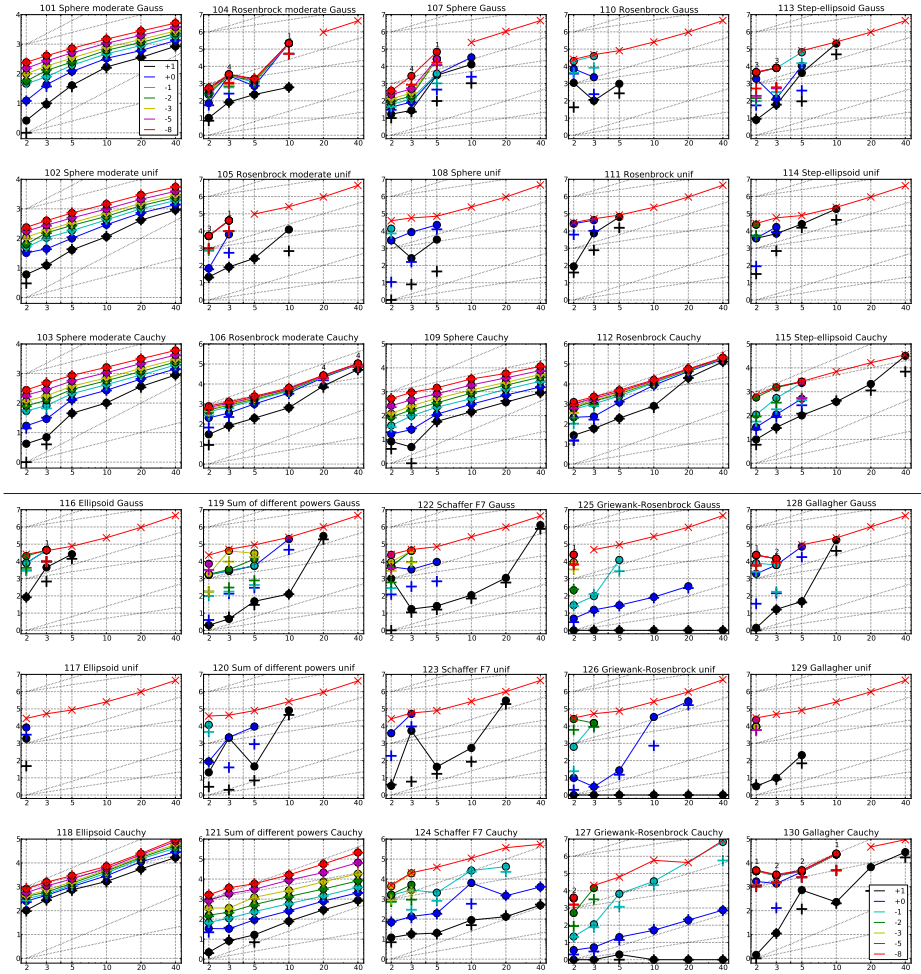


Figure 4: Expected Running Time (ERT, ●) to reach $f_{\text{opt}} + \Delta f$ and median number of function evaluations of successful trials (+) for CMA-ES with standard population size, shown for $\Delta f = 10, 1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-5}, 10^{-8}$ (the exponent is given in the legend of f_{101} and f_{130}) versus dimension in log-log presentation. The ERT(Δf) equals to #FEs(Δf) divided by the number of successful trials, where a trial is successful if $f_{\text{opt}} + \Delta f$ was surpassed during the trial. The #FEs(Δf) are the total number of function evaluations while $f_{\text{opt}} + \Delta f$ was not surpassed during the trial from all respective trials (successful and unsuccessful), and f_{opt} denotes the optimal function value. Crosses (×) indicate the total number of function evaluations #FEs($-\infty$). Numbers above ERT-symbols indicate the number of successful trials. Annotated numbers on the ordinate are decimal logarithms. Additional grid lines show linear and quadratic scaling.

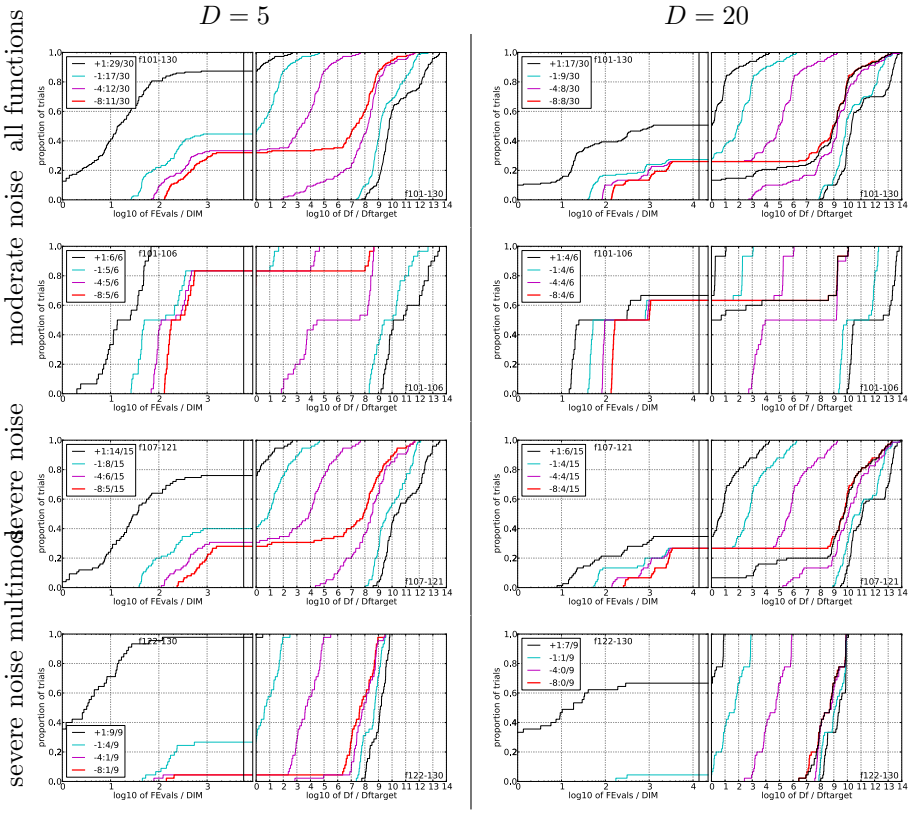


Figure 5: Empirical cumulative distribution functions (ECDFs) for CMA-ES with standard population size, plotting the fraction of trials versus running time (left subplots) or versus Δf (right subplots). The thick red line represents the best achieved results. Left subplots: ECDF of the running time (number of function evaluations), divided by search space dimension D, to fall below $f_{\mathrm{opt}}+\Delta f$ with $\Delta f = 10^k$, where k is the first value in the legend. Right subplots: ECDF of the best achieved Δf divided by 10^k (upper left lines in continuation of the left subplot), and best achieved Δf divided by 10^{-8} for running times of $D, 10\,D, 100\,D\ldots$ function evaluations (from right to left cycling black-cyan-magenta). Top row: all results from all functions; second row: moderate noise functions; third row: severe noise functions; fourth row: severe noise and highly-multimodal functions. The legends indicate the number of functions that were solved in at least one trial. FEvals denotes number of function evaluations, D and DIM denote search space dimension, and Δf and Df denote the difference to the optimal function value.

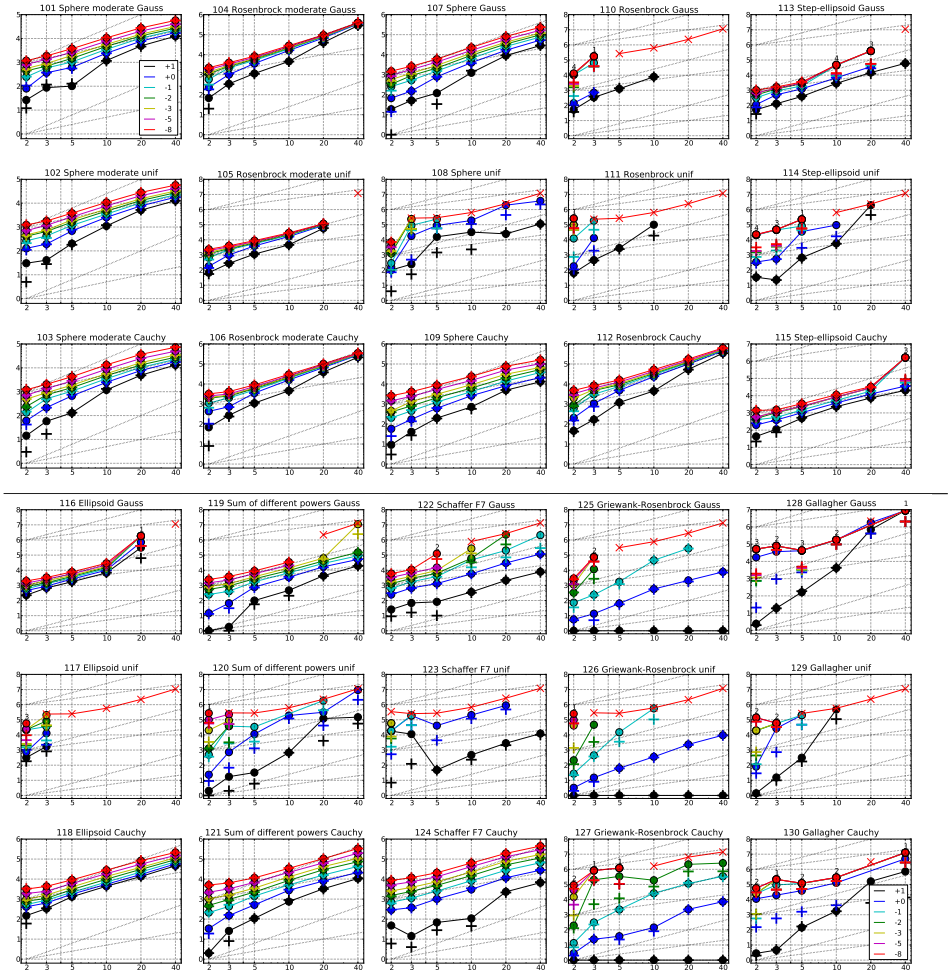


Figure 6: Expected Running Time (ERT, ●) to reach $f_{\text{opt}} + \Delta f$ and median number of function evaluations of successful trials (+) for CMA-ES with population size 16 times standard, shown for $\Delta f = 10, 1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-5}, 10^{-8}$ (the exponent is given in the legend of f_{101} and f_{130}) versus dimension in log-log presentation. The ERT(Δf) equals to #FEs(Δf) divided by the number of successful trials, where a trial is successful if $f_{\text{opt}} + \Delta f$ was surpassed during the trial. The #FEs(Δf) are the total number of function evaluations while $f_{\text{opt}} + \Delta f$ was not surpassed during the trial from all respective trials (successful and unsuccessful), and f_{opt} denotes the optimal function value. Crosses (×) indicate the total number of function evaluations #FEs($-\infty$). Numbers above ERT-symbols indicate the number of successful trials. Annotated numbers on the ordinate are decimal logarithms. Additional grid lines show linear and quadratic scaling.

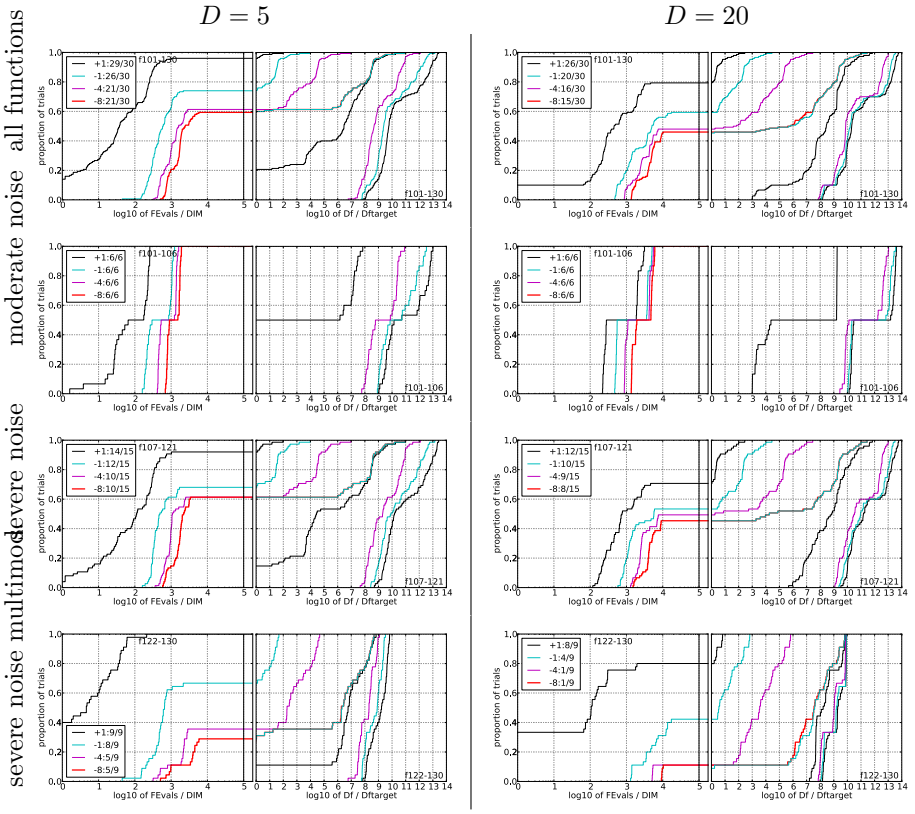


Figure 7: Empirical cumulative distribution functions (ECDFs) for CMA-ES with population size 16 times standard, plotting the fraction of trials versus running time (left subplots) or versus Δf (right subplots). The thick red line represents the best achieved results. Left subplots: ECDF of the running time (number of function evaluations), divided by search space dimension D, to fall below $f_{\mathrm{opt}} + \Delta f$ with $\Delta f = 10^k$, where k is the first value in the legend. Right subplots: ECDF of the best achieved Δf divided by 10^k (upper left lines in continuation of the left subplot), and best achieved Δf divided by 10^{-8} for running times of $D, 10\,D, 100\,D \ldots$ function evaluations (from right to left cycling black-cyan-magenta). Top row: all results from all functions; second row: moderate noise functions; third row: severe noise functions; fourth row: severe noise and highly-multimodal functions. The legends indicate the number of functions that were solved in at least one trial. FEvals denotes number of function evaluations, D and DIM denote search space dimension, and Δf and Df denote the difference to the optimal function value.

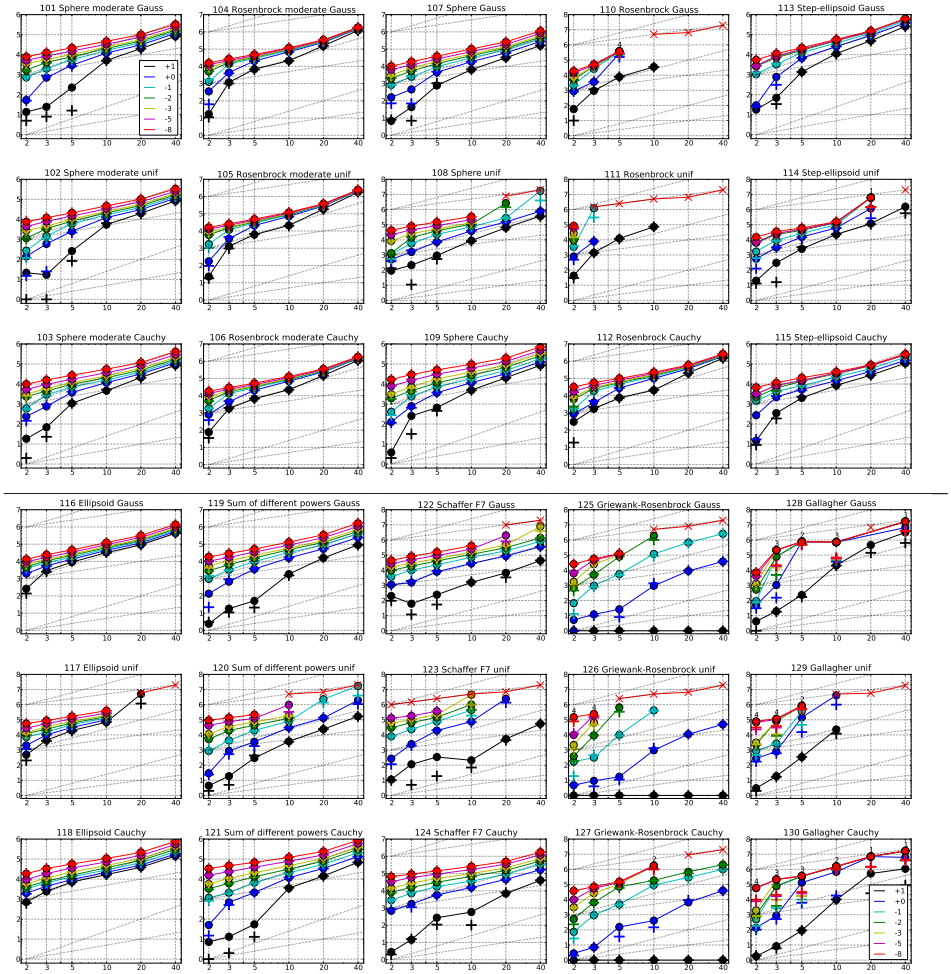


Figure 8: Expected Running Time (ERT, ●) to reach $f_{\text{opt}} + \Delta f$ and median number of function evaluations of successful trials (+) for CMA-ES with population size 128 times standard, shown for $\Delta f = 10, 1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-5}, 10^{-8}$ (the exponent is given in the legend of f_{101} and f_{130}) versus dimension in log-log presentation. The ERT(Δf) equals to #FEs(Δf) divided by the number of successful trials, where a trial is successful if $f_{\text{opt}} + \Delta f$ was surpassed during the trial. The #FEs(Δf) are the total number of function evaluations while $f_{\text{opt}} + \Delta f$ was not surpassed during the trial from all respective trials (successful and unsuccessful), and f_{opt} denotes the optimal function value. Crosses (×) indicate the total number of function evaluations #FEs($-\infty$). Numbers above ERT-symbols indicate the number of successful trials. Annotated numbers on the ordinate are decimal logarithms. Additional grid lines show linear and quadratic scaling.

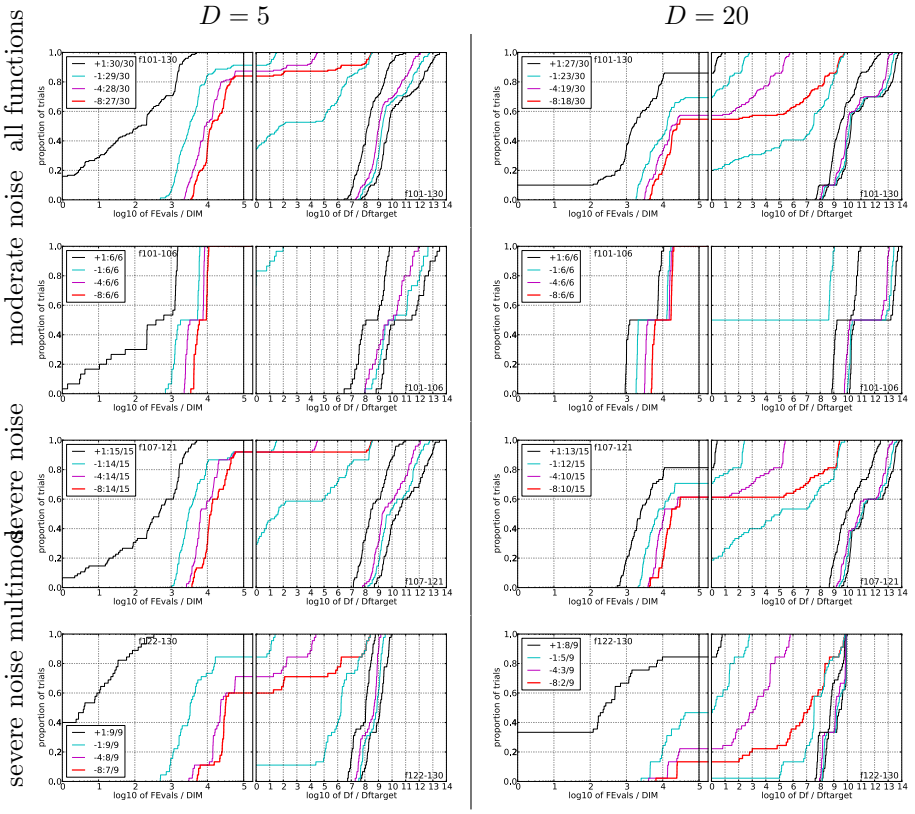


Figure 9: Empirical cumulative distribution functions (ECDFs) for CMA-ES with population size 128 times standard, plotting the fraction of trials versus running time (left subplots) or versus Δf (right subplots). The thick red line represents the best achieved results. Left subplots: ECDF of the running time (number of function evaluations), divided by search space dimension D, to fall below $f_{\mathrm{opt}} + \Delta f$ with $\Delta f = 10^k$, where k is the first value in the legend. Right subplots: ECDF of the best achieved Δf divided by 10^k (upper left lines in continuation of the left subplot), and best achieved Δf divided by 10^{-8} for running times of $D, 10\,D, 100\,D \ldots$ function evaluations (from right to left cycling black-cyan-magenta). Top row: all results from all functions; second row: moderate noise functions; third row: severe noise functions; fourth row: severe noise and highly-multimodal functions. The legends indicate the number of functions that were solved in at least one trial. FEvals denotes number of function evaluations, D and DIM denote search space dimension, and Δf and Df denote the difference to the optimal function value.

the corresponding objective function value is not computed during the CMA-ES run but has been obtained by a subsequent re-evaluation of the parameters. The re-evaluated objective function values of both $\vec{x}_{\text{best}}$ and $\vec{x}_{\text{mean}}$ can be found in Table 2.

Interesting is that we observe for two runs that the overall best solution, when re-evaluated, shows a worse performance than the solution $\vec{x}_{\text{mean}}$ in the end of the optimization (Table 2). That the re-evaluation of a point changes the result in practice (although it should be the same due to the fixed random seed) let us come to the conclusion that numerical errors are responsible for the large variation in the observed objective function values which we called *noise* before.

9 Re-Evaluating Points Around the Optimum

After also evaluating some randomly generated solutions around the found solutions of Table 2 that were created by adding a normally distributed random vector (mean 0, variance 10^{-16}) showed almost always the same objective function value, our conclusion that numerical errors are responsible for the behavior of CMA-ES became more founded.

To figure out to which amount the numerical errors affect the objective function and thereby the optimization with CMA-ES, we evaluated further points on a line through the search space defined by the (approximative) implanted optimum and a random direction[7]. Figure 10 shows the resulting objective function values for two random directions. Both plots show the same two main characteristics of the objective function: (i) overall, although the objective function is noisy, the objective function values increase with larger distance to the optimum (indicating an almost unimodal function with additional noise) and (ii) and more importantly, there are large plateaus close to the optimum (starting from about a distance of 10^{-7} to 10^{-8} from the optimum) the objective function value of which are *not* close to the optimal value possible and that are also not directing the search towards the optimum.

Both observations coincide with what we have seen on the CMA-ES output, e.g., in Fig. 2: First, CMA-ES sees the course-grained view of the objective function: closer to the optimum, better function values can be found, and the current fitness decreases (behavior within the first about 1000 function evaluations). Then, while decreasing its step size, CMA-ES enters the region of high noise (distance of about 10^{-2} to 10^{-7} to the optimum). The current objective function value varies highly (> 1000 function evaluations in Fig. 2). Due to the plateaus of equal, but high function values around the optimum, CMA-ES is not able to converge closer to the optimum than with a precision of about 10^{-7}.

To conclude, with the current formalization and implementation of the problem, not enough information about the optimum is contained in the objective function such that CMA-ES is not able to converge closer than 10^{-7} to the optimum. However, CMA-ES can find parameter values close to the desired once which might be already precise enough in practice where the objective function

[7] Note that also the re-evaluation of the optimal parameter values that should theoretically result in an objective function of 0 showed a non-zero objective function value due to the numerical errors—the reason why we used the term *approximative optimum* here.

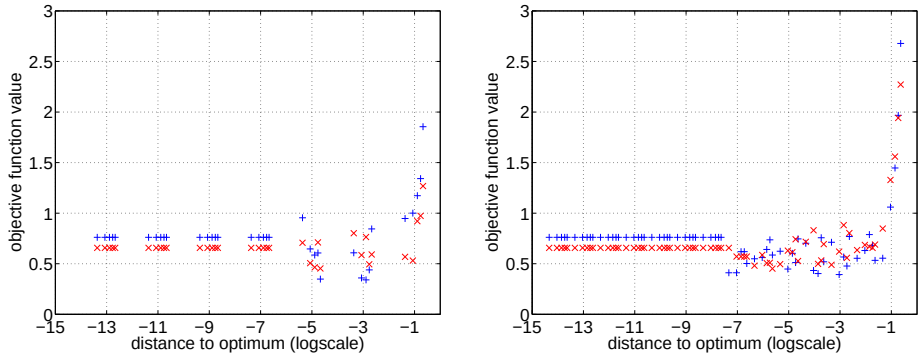


Figure 10: Fitness function sampled on a line opt + td for two random directions d = (0.754686681982361, 0.276025076998578, 0.679702676853675, 0.655098003973841, 0.162611735194631, 0.118997681558377) (left) and d = (0.278498218867048, 0.546881519204984, 0.957506835434298, 0.964888535199277, 0.157613081677548, 0.970592781760616) (right). The distance to the optimum is given in logarithmic scale, i.e., a value of i on the x-axis corresponds to a distance of 10^i to the optimum. Solutions in direction of d are drawn as 'x' whereas a '+' denotes a solution lying in direction $-d$ from the optimum.

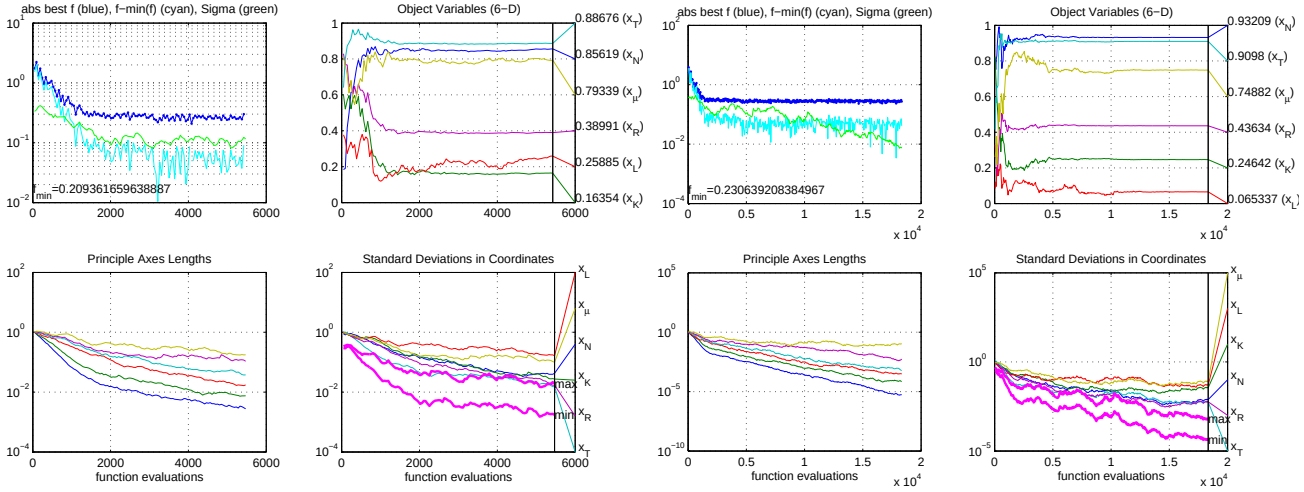


Figure 11: Standard output of the CMA-ES for two independent runs with a population size of $\lambda = 48$. Left: first run with 5424 function evaluations; Right: second run with 18226 function evaluations. The reason for the early stop of the run shown on the left was the meeting of the stopping criterion *optimal value found*. However, we believe that this was caused by a numerical error as the re-evaluation of the corresponding search point gave an objective value of 0.4116519215632899.

will depend on real-world, and therefore noisy measurements which makes an exact parameter estimation (in a mathematical sense of convergence) needless.

popsize (λ)	generations	#fitevals	date	description and characteristics
24	367	8808	18/11/2009	first run with reasonable CMA settings
48	113	5424	20/11/2009	automatic stop as generation 114 caused fitness of exactly 0
48	382	18336	19/12/2009	
100	115	11500	03/01/2010	
200	41	8200	10/01/2010	crash due to bug with MySQL connection
200	126	25200	23/01/2010	crash due to re-evaluation code that was run in parallel
9	1000	9000	27/01/2010	automatic stop after 1000 generations

Table 1: An overview of all simulation runs performed at the letsim server.

description of run	$f(\vec{x}_{\text{best}})$	$f(\vec{x}_{\text{mean}})$
$\lambda = 24$	0.2288760783901613	0.4481500612141598
$\lambda = 48$ (1st run)*	0.20936165963888653	0.3486784279110696
$\lambda = 48$ (2nd run)	0.8283756851333607	0.7641699819522306
$\lambda = 100$	0.7881974016913034	0.6820863358456356
$\lambda = 200$ (1st run)	0.6929446767981414	0.8223911237931907
$\lambda = 200$ (2nd run)	0.20057145725406839	0.4479376610066943
$\lambda = 9$	0.2189927371627679	0.4575498353751773

Table 2: Re-evaluated objective function values for the best solution $\vec{x}_{\text{best}}$ found in each calibration run as well as re-evaluated objective values for $\vec{x}_{\text{mean}}$ of the last generation. *For $\vec{x}_{\text{best}}$, the best solution in generation 85 has been used here where the solution in the very last generation resulting in CMA-ES to stop due to a fitness of 0 has been re-evaluated with an objective function value of 0.4116519215632899.

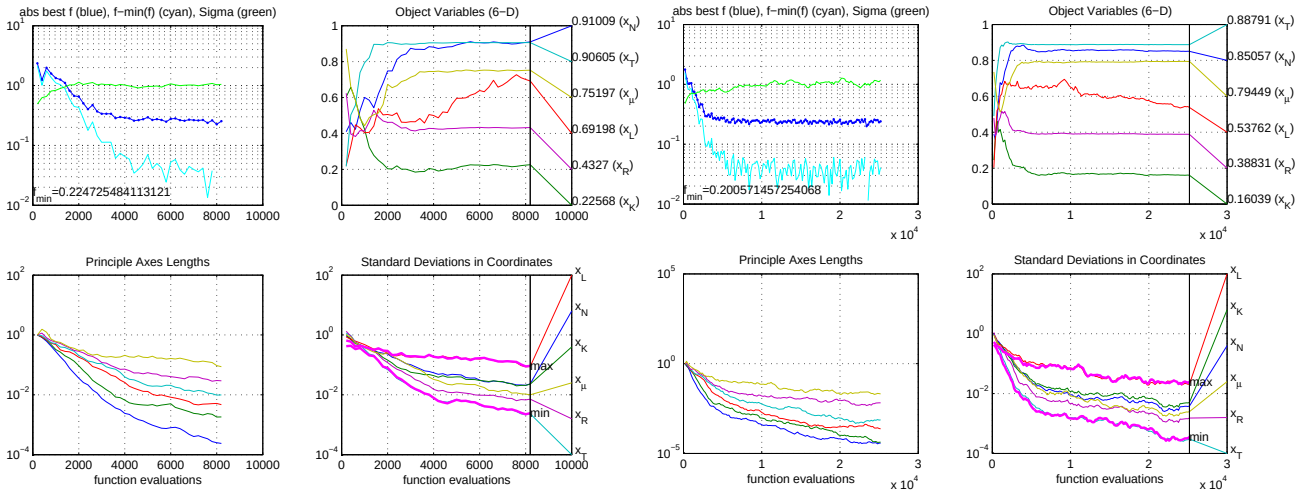


Figure 12: Standard output of the CMA-ES for two independent runs with a population size of $\lambda = 200$. Left: first run with 8200 function evaluations; Right: second run with 25200 function evaluations.

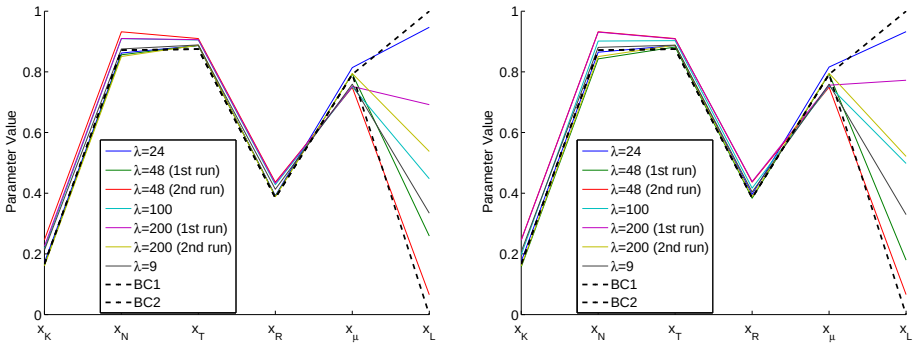


Figure 13: Parallel coordinates plot for the parameter values found by CMA-ES in terms of the $\vec{x}_{\text{mean}}$ in the last generation (left) and the overall best $\vec{x}_{\text{best}}$ (right) for all runs of Table 1. In addition, the parameter values corresponding to the scenario implanted in the objective function are plotted as a dashed thick line where x_L is arbitrary.

10 Conclusion

When simulation output and real data have to be matched, e.g., when predicting car traffic, calibration of both the model's and the simulator's parameters are crucial. In this paper, we provide a first preliminary study about how this can be achieved with a stochastic optimization algorithm called CMA-ES (Covariance Matrix Adaptation Evolution Strategy) for the mesoscopic car traffic simulator Metropolis. As calibrating the traffic simulation is computationally expensive, the evaluations of the six-dimensional objective function have been parallelized on a grid of up to 44 cores. The preliminary results are twofold: On the one hand, we could show that the optimization is possible, i.e., the desired parameter values could be found. On the other hand, the objective function turned out to be highly noisy or more correctly rugged as the random seed of the simulator's pseudorandom number generator has been fixed. This noise was the main reason why CMA-ES could not converge log-linearly as usually observed on many (also noisy) test functions.

The second part of the paper was therefore trying to investigate this noise as well as CMA-ES's capability to deal with it further. Benchmarking experiments of CMA-ES on 30 noisy test functions suggested that increasing the algorithm's population size increases the ability of the algorithm to find solutions with smaller and smaller objective function values. However, this impact of the population size could not be observed with the calibration problem and re-evaluating points suggested that numerical errors prevent CMA-ES from approaching the optimum with arbitrary precision. Hence, no further improvement of the found solutions can be expected in the current scenario.

Based on these foundings about calibrating the Metropolis simulator, the next steps would be to increase the number of parameters (e.g. optimizing also the entries of the O-D-matrix) as well as the incorporation of real-world data in the optimization process.

Acknowledgments

This work has been supported by the French national research agency (ANR) within the SYSCOMM project ANR-08-SYSC-017.

References

[1] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler. PISA—A Platform and Programming Language Independent Interface for Search Algorithms. In C. M. Fonseca et al., editors, *Conference on Evolutionary Multi-Criterion Optimization (EMO 2003)*, volume 2632 of *LNCS*, pages 494–508, Berlin, 2003. Springer.

[2] A. de Palma and F. Marchal. Real Cases Applications of the Fully Dynamic METROPOLIS Tool-Box: An Advocacy for Large-Scale Mesoscopic Transportation Systems. *Networks and Spatial Economics*, 2:347–369, 2002.

[3] A. de Palma, F. Marchal, and Y. Nesterov. METROPOLIS: Modular System for Dynamic Traffic Simulation. *Transportation Research Record: Journal of the Transportation Research Board*, 1607:178–184, 1997.

[4] S. Finck and H. G. Beyer. Benchmarking SPSA on BBOB-2010 Noiseless Function Testbed. In *GECCO workshop on Black-Box Optimization Benchmarking (BBOB'2010)*. ACM, 2010. to appear.

[5] S. Finck and H. G. Beyer. Benchmarking SPSA on BBOB-2010 Noisy Function Testbed. In *GECCO workshop on Black-Box Optimization Benchmarking (BBOB'2010)*. ACM, 2010. to appear.

[6] N. Hansen. Benchmarking a BI-Population CMA-ES on the BBOB-2009 Function Testbed. In *Workshop proceedings of the GECCO workshop on blackbox Optimization Benchmarking (BBOB 2009)*, pages 2389–2395. ACM, 2009.

[7] N. Hansen, A. Auger, S. Finck, and R. Ros. Real-Parameter Black-Box Optimization Benchmarking 2009: Experimental Setup. INRIA Research Report RR-6828, INRIA Saclay—Ile-de-France, May 2009.

[8] N. Hansen and S. Kern. Evaluating the CMA Evolution Strategy on Multimodal Test Functions. In X. Yao et al., editors, *Conference on Parallel Problem Solving from Nature (PPSN VIII)*, volume 3242 of *LNCS*, pages 282–291, Berlin, Germany, 2004. Springer.

[9] D. A. Hensher and K. J. Button, editors. *Handbook of Transport Modelling*. Elsevier, 2000.

[10] L. J. Le Blanc, E. K. Morlok, and W. P. Pierskalla. An Efficient Approach to Solving the Road Network Equilibrium Traffic Assignment Problem. *Transportation Research*, 9:309–318, 1975.

[11] J. Medina, M. Moreno, and E. Royo. Evolutionary Computation Applied to Urban Traffic Optimization. In W. Kosiński, editor, *Advances in Evolutionary Algorithms*, pages 421–442. I-Tech Education and Publishing, Vienna, Austria, 2008.

[12] N. Melab, S. Cahon, and E.-G. Talbi. Grid Computing for Parallel Bioinspired Algorithms. *Journal of Parallel and Distributed Computing*, 66:1052–1061, 2006.

[13] C. L. Müller, B. Baumgartner, G. Ofenbeck, B. Schrader, and I. F. Sbalzarini. pCMALib: a Parallel FORTRAN 90 Library for the Evolution Strategy with Covariance Matrix Adaptation. In G. Raidl et al., editors, *Genetic and Evolutionary Computation Conference (GECCO 2009)*, pages 1411–1418, New York, NY, USA, 2009. ACM.

[14] J. Sánchez, M. Galán, and E. Rubio. Applying a Traffic Lights Evolutionary Optimization Technique to a Real Case: "Las Ramblas" Area in Santa Cruz de Tenerife. *IEEE Transactions on Evolutionary Computation*, 12(1):25–40, 2008.

[15] J. Spall. Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341, 1992.

[16] V. Vaze. Calibration of Dynamic Traffic Assignment Models with Point-to-Point Traffic Surveillance. Master's thesis, Massachusetts Institute of Technology, 2007.

[17] A. Vogel, C. Goerick, and W. von Seelen. Evolutionary Algorithms for Optimizing Traffic Signal Operation. In *European Symposium on Intelligent Techniques (ESIT 2000)*, pages 83–91, 2000.

Centre de recherche INRIA Saclay – Île-de-France
Parc Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 Orsay Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399