Vrije Universiteit

Research Paper

Business Analytics

Sample-Efficient Optimizers: Bayesian Optimization vs. CMA-ES

Author:
Daan Crebolder

Supervisor:
Prof. Dr. A.E. Eiben

December 17, 2018

Abstract

In the application of evolutionairy robotics, robots need to learn to locomote in as few trials as possible. Therefore, a suitable optimization strategy is one that causes big improvement in as few iterations as possible, but not necessarily finds the global optimum in the long run. This paper compares the sample-efficiency of Bayesian Optimization (BO) and Covariance Matrix Adaptation Evolution Strategy (CMA-ES). Both strategies are tested on a set of noise free and noisy black-box optimization functions. This research concludes that BO outperforms CMA-ES on this criterion, but is less reliable and computationally more intensive.

Contents

1 Introduction

In recent years, evolutionary computing has expanded, from simulations, into the domain actual robot babies. The robots' morphologies are physically constructed and their fitness is evaluated by measuring the distance they are able to locomote in, say, a minute. In this process, each morphology is unique, as is its controller. Therefore, the robots need to learn to locomote themselves, i.e. optimize their controller.

Compared to running an evolutionary algorithm by simulation, the process of evaluating physical robots is time consuming and expensive. Therefore, the learner/optimizer of the robots' controllers should prioritize efficiency over efficacy. In other words, we would prefer an optimizer that makes the robots learn to move decently as soon as possible, over one that takes more time but makes makes the robot move perfectly eventually. We will call such an optimizer sample-efficient.

In this paper we compare the sample-efficiency of two optimizers: Bayesian optimization (BO) and CMA-ES. We also compare their resilience to noise, and discuss which would be best suited for robotics.

Their performances are not tested on real robots but on a test-suite consisting of a selection of noiseless and noisy black-box optimization functions. The problem that needs to be solved can simply be written as

$$\min_{\mathbf{x}\in[-5,5]^d} f(\mathbf{x})$$

where $f(\mathbf{x})$ is an unknown objective function, $\mathbf{x}$ the input vector, and d the dimension of the input vector for which we will use $\{2, 5, 10, 20, 40\}$.

This paper first gives a brief overview of the two optimizers and the test-suite and then discusses which of the two we find best suited for evolutionary robotics.

2 Background

2.1 Bayesian Optimization

The fundamental characteristic of Bayesian Optimization is that it uses all the prior knowledge available to estimate which next point is most suitable for evaluation. If, for example, an unknown objective function $f(\mathbf{x})$ is evaluated at multiple points $\mathbf{x}$, it guesses the shape and the uncertainty of $f(\mathbf{x})$ at not evaluated points. Based on this prior distribution on $f(\mathbf{x})$, also called the surrogate model, it chooses at which point the function is evaluated next, taking into account the expected value and the uncertainty.

Figure 1 shown an illustration of the algorithm on a case with a one dimensional input value. The algorithm starts with $d + 1$ randomly chosen evaluated points, what we will call the initial design. The dashed line is the unknown objective function $f(\mathbf{x})$. The black dots are previous evaluations. The solid black line and the blue area represent the mean of the surrogate and its uncertainty. The next point to be evaluated, the red dot, it chosen at some combination of minimum value and high uncertainty. This combination is quantified by an acquisition function, represented by the green line. At every iteration t en new point is evaluated, a new surrogate and acquisition function are constructed and a next point is chosen to evaluate in $t + 1$. The prior of iteration $t + 1$ can be seen as the posterior of iteration t.

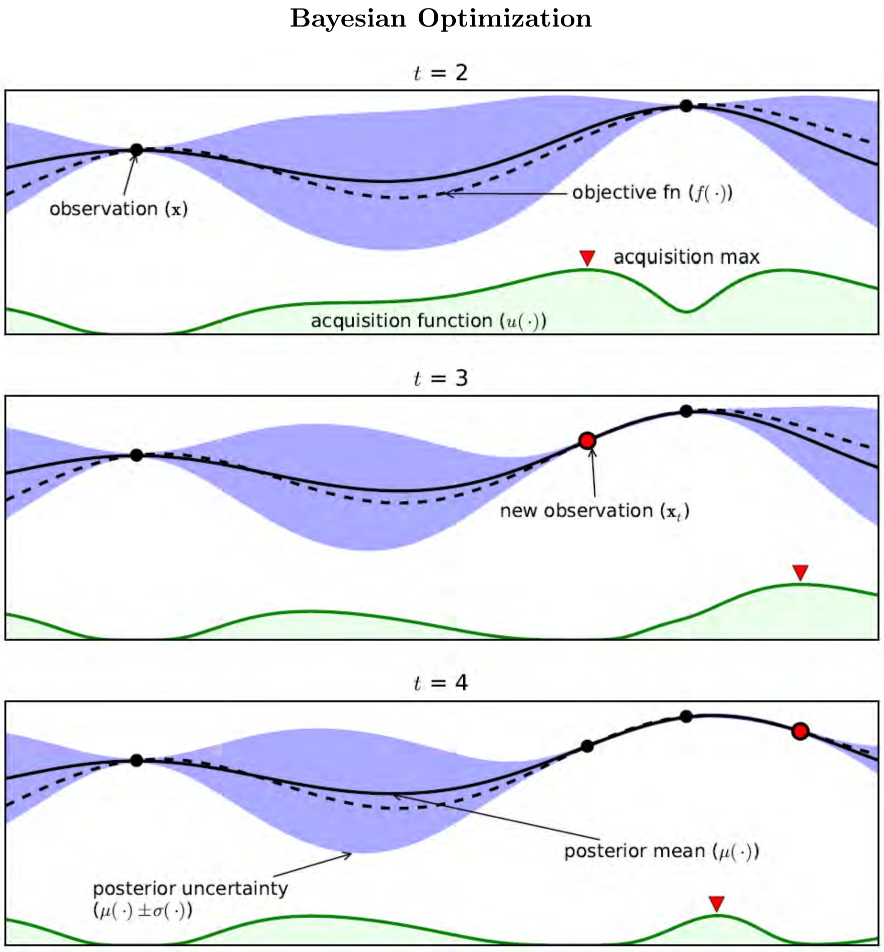


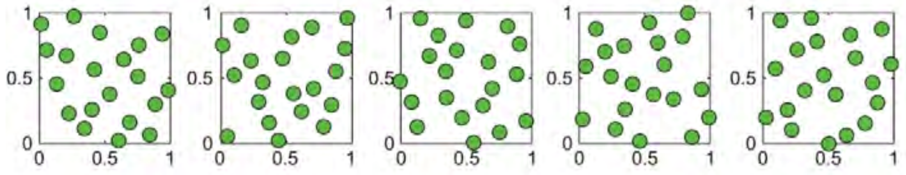
Figure 1: An illustration of Bayesian Optimization[1]

The next sections will discuss the initial design, the construction of the prior using a Gaussian process and possible acquisition functions.

2.1.1 Initial design

In order to construct a prior, first, at least d points have to be known, or evaluated. These points can be drawn randomly form the domain of the function, or quasi randomly to avoid the points being close together. Maximin LHS, for example, maximizes the minimal distance between the points. It randomly creates a large number of designs, like in figure 2.1.1, and chooses the one satisfying the function:

$$M(\mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^n) = min_{i,j} \parallel \mathbf{x}_i - \mathbf{x}_j \parallel$$



2.1.2 Construction the surrogate

The prior, or surrogate, over the unknown objective function serves as an estimation the that function. In the scope of this research the prior is made using a Gaussian Process (GP). For every point of the objective function, the GP defines a normal distribution with a mean and variance. Over the whole function the GP therefore gives a mean function and a covariance function k.

$$f(\mathbf{x}) \sim GP(\mu(\mathbf{x}), k(f(\mathbf{x}), f(\mathbf{x})'))$$

A covariance function assigns the covariance of points $\in [0,1]$ depending on how close they are together. At nearby points the values of the objective function are assumed more similar than at distant points. There are multiple choices for the covariance function. A simple and popular one is the squared exponential function:

$$k(\mathbf{x}_i, \mathbf{x}_j) = exp(-\frac{1}{2} \parallel \mathbf{x}_i, \mathbf{x}_j \parallel^2)$$

A more flexible one is the Matèrn function:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{2^{\zeta-1}\Gamma(\zeta)}(2\sqrt{\zeta} \parallel \mathbf{x}_i - \mathbf{x}_j \parallel)^{\zeta} H_{\zeta}(2\sqrt{\zeta} \parallel \mathbf{x}_i - \mathbf{x}_j \parallel)$$

with $\Gamma()$ the Gamma function, H_ζ the Bessel function and ζ a parameter. Note that as $\zeta \to \infty$, the Matèrn function goes to the squared exponential function. If $\zeta = 0.5$ it is the unsquared exponential function.

As explained in Brochu et al. (2010)[1], the prior can be constructed as follows. If the kernel matrix $\mathbf{K}$ is defined as

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \dots & k(\mathbf{x}_1, \mathbf{x}_t) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_t, \mathbf{x}_1) & \dots & k(\mathbf{x}_t, \mathbf{x}_t) \end{bmatrix}$$

at iteration t the function value $\mathbf{f}_{t+1}$ at a next unknown point $\mathbf{x}_{t+1}$ given the information on previously evaluated function values $\mathbf{f}_{1:t}$ can be estimated as

$$\begin{bmatrix} \mathbf{f}_{1:t} \\ \mathbf{f}_{t+1} \end{bmatrix} \sim \mathcal{N}\left(\mu, \begin{bmatrix} \mathbf{K} & \mathbf{k} \\ \mathbf{k}^T & k(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) \end{bmatrix}\right)$$

where $\mathbf{k} = [k(\mathbf{x}_{t+1}, \mathbf{x}_1), k(\mathbf{x}_{t+1}, \mathbf{x}_2) \dots k(\mathbf{x}_{t+1}, \mathbf{x}_t)]$

Which can be derived to

$$P(f_{t+1} \mid D_{1:t}, \mathbf{x}_{t+1}) = \mathcal{N}(\mu_t(\mathbf{x}_{t+1}), \sigma_t^2(\mathbf{x}_{t+1}))$$

Noise can be accounted for by simply adding $\sigma^2_{noise} I$ to $\mathbf{K}$ which results in

$$P(f_{t+1} \mid D_{1:t}, \mathbf{x}_{t+1}) = \mathcal{N}(\mu_t(\mathbf{x}_{t+1}), \sigma_t^2(\mathbf{x}_{t+1})) + \sigma^2_{noise}$$

2.1.3 Acquisition functions

After a prior is constructed, a next point $\mathbf{x}_{t+1}$ has to be chosen where the objective function will be evaluated in the next iteration. In this choice there is a trade-off between exploitation and exploration. Exploitation in this process entails that we prefer a point where the mean of the GP is minimal, the black line in figure 1, where we expect to find a low function value. Exploration entails a preference for a point where the covariance of the GP is high, or where the blue area in figure 1 is tall. An evaluation at such a point will give us a more informative posterior. An acquisition function, the green line in figure 1, contains a mixture of low mean and high covariance of the GP. It should be maximized to find $\mathbf{x}_{t+1}$.

An example of an acquisition function is the Lower Confidence Bound (LCB). For maximization problems this is called the Upper Confidence Bound.

$$\mathbf{x}_{t+1} = \arg\min_{\mathbf{x}}(\mu_t(\mathbf{x}) + \kappa\sigma_t(\mathbf{x}))$$

the parameter $\kappa \in [0, \infty)$ controls the exploration/exploitation trade-off. This parameter can change during a run of the algorithm, as it might be useful to focus on exploration in the first few iteration and more on exploitation in later iterations.

2.2 CMA-ES

Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is an algorithm that adapts the covariance matrix of a multi-variate normal search distribution. It is well suited for optimizing real-valued complex functions [2]. It is designed to perform well for small populations and might therefore be considered sample-efficient. Each iteration, a population is drawn form a distribution which maximized the likelihood of solutions with a high fitness value. This distribution is updated every iteration by adapting its covariance matrix. This strategy is meant to converge quickly while preventing premature convergence.

Figure 2 shows an illustration of the algorithm for a two-dimensional problem. Each generation the distribution from which the points are sampled move towards the global optimum.

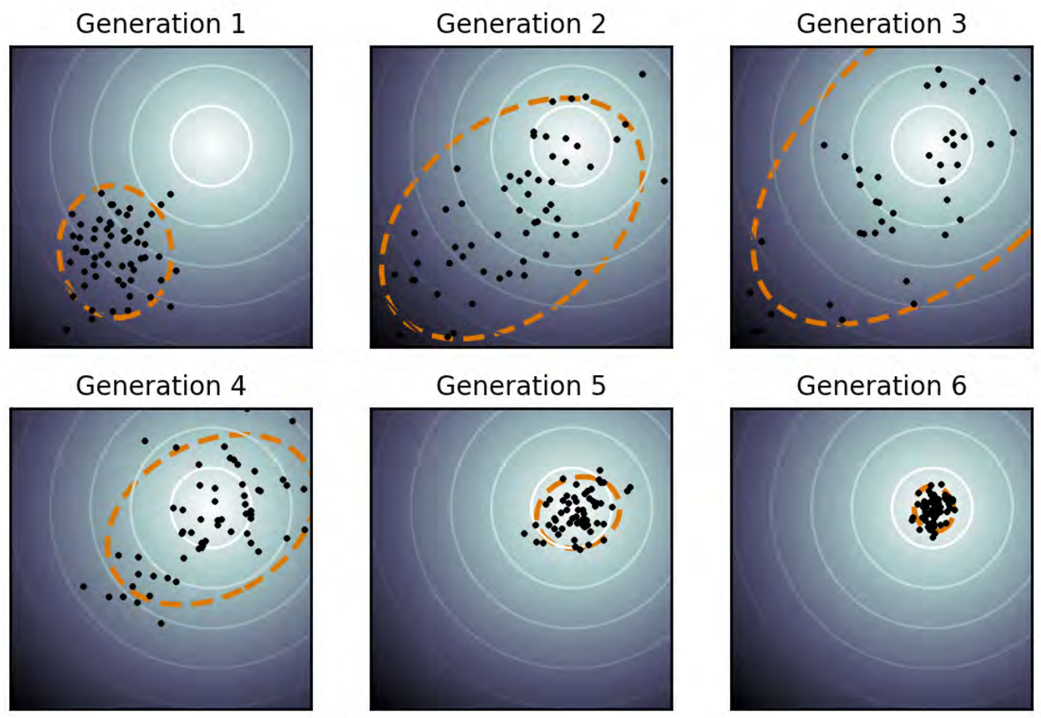


Figure 2: Six generations of a CMA-ES optimization of a two-dimensional problem.

3 Methods

3.1 Test suite

The optimizers will be evaluated with the COCO (Comparing Continuous Optimizers) platform, which has been used for the Black-Box-Optimization-Benchmarking (BBOB) workshops during GECCO conferences in several years, as well as for the Congress on Evolutionary Computation (CEC) in 2015. We will use the nine BBOB benchmark functions described in this chapter, as presented by Flinck et al (2009) [3][4]. The numbering of the functions (e.g. f_{12}) corresponds to the BBOB numbering.

In choosing these nine functions we mainly aim to create a diverse test suite, meaning that the functions are of differing complexities and have varying shapes. The types of noise and noise intensities are also meant to vary. The noisy functions each have a noiseless counterpart, in order to analyze the effect of the noise. Because the test suite consists of diverse functions, the behavior of the optimizers can be better understood.

The nine functions are plotted in figure 3 with a dimensionality of 2. With more dimensions they become harder to visualize but their distinctive features remain. Note that the f-axes are upside down. All function are to be minimized, so the optima are peaks.

At each instance of any of the functions, it is randomly shifted in both the x-space and the f-space. This causes the location and the value of the optimum to be unpredictable.

- f_{opt} : optimal function value.
- $D = [10, 20, 30, 40, 50]$, search space dimensionality.
- The domain of all functions is $\mathcal{R}^D$ and they have their optimum on $[-5, 5]^D$.
- $f_{pen} : \mathcal{R}^D \rightarrow \mathcal{R}, x \mapsto \sum_{i=1}^{D} max(0, |x_i| - 5)^2$, a penalty for solutions outside the domain.
- $\wedge^\alpha$ is a diagonal matrix in D dimensions with the ith diagonal element as $\lambda_{ii} = \alpha^{\frac{1}{2}\frac{i-1}{D-1}}$, for $i = 1, ..., D$.
- Q and R are orthogonal matrices.
- On simpler functions a non-linear transformation is applied to make them less regular: $T^\beta_{asy} : \mathcal{R}^D \rightarrow \mathcal{R}^D, x_i \mapsto x_i^{1+\beta\frac{i-1}{D-1}\sqrt{x_i}}$ if $x > 0$.

3.1.1 Noiseless Functions

f_1 **Sphere Function** The unimodal sphere is a simple function whose optimum is probably quickly found. Still, including it in our test suite could be useful for obtaining the optimizers optimal convergence rate. This result can be compared with more complex functions.

$$f_1 = ||z||^2 + f_{opt}$$

$z = x - x^{opt}$

f_8 **Rosenbrock** The Rosenbrock or banana function is interesting, especially in larger dimensions, because the slope leading to the optimum changes direction D-1 times.

$$f_8(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_{opt}$$

$z = \max(1, \frac{\sqrt{D}}{8})(x - x^{opt}) + 1$ $z^{opt} = 1$

f_{17} **Schaffers F7** The Schaffers F7 function has many peaks of varying density and intensity. Since the frequency and amplitude of the modulation vary, with this function it can be tested how well an optimizer can distuinguish the global optimum from local optima.

$$f_{17}(x) = \left(\frac{1}{D-1}\sum_{i=1}^{D-1} \sqrt{s_i} + \sqrt{s_i}\sin^2\left(50 s_i^{1/5}\right)\right)^2 + 10 f_{pen}(x) + f_{opt}$$

$z = \wedge^{10} Q T^{0.5}_{asy}(R(x - x^{opt}))$
$s_i = \sqrt{z_i^2 + z_{i+1}^2 + ...}$ for $i = 1, ..., D$

f_{21} **Gallagher's Gaussian 101-me Peaks** Each instantiation of Gallagher's Gaussian 101-me Peaks consists of 101 randomly chosen peaks. With this function it can be tested how the optimizers deal with functions without any global structure.

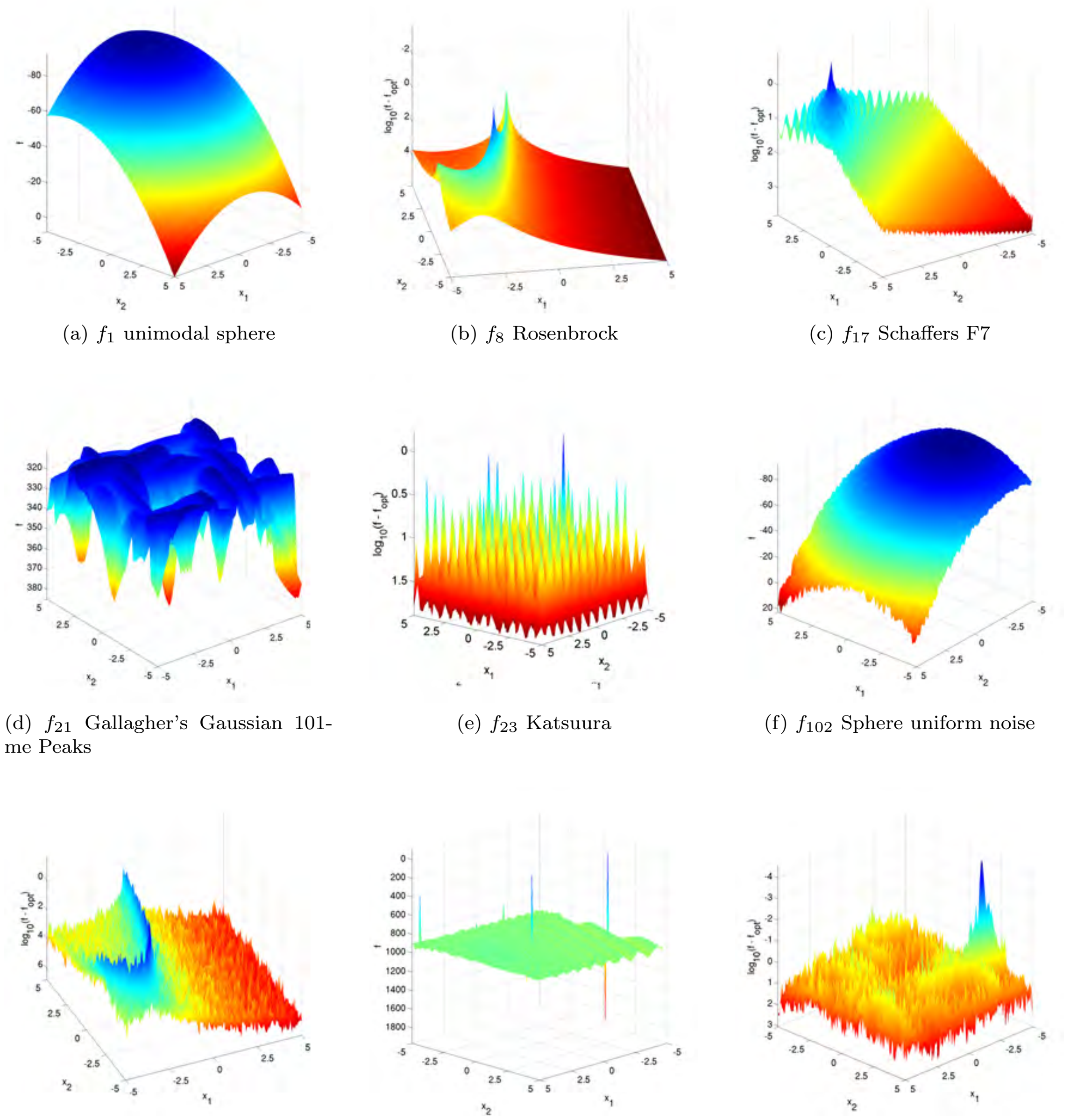
$$f_{21}(x) = T_{osz}\left(10 - \max_{i=1}^{101} w_i \exp\left(-\frac{1}{2D}(x - y_1)^T R^T C_i R(x - y_i)\right)\right)^2 + f_{pen}(x) + fopt$$

$w_i = 1.1 + 8\frac{i-2}{99}$ for $i = 2, ..., 101$ and $w_i = 10$ for $i = 1$
$C_i = \wedge^{\alpha_i} / \alpha_i^{1/4}$

f_{23} **Katsuura** The Katsuura function has an obvious pattern with local optima equally spaced from each other. This function is highly rugged and highly repetitive, so it has multiple global optima.

$$f_{23}(x) = \frac{10}{D^2} \prod_{i=1}^{D} \left(1 + i \sum_{j=1}^{3} 2 \frac{|2^j z_i - [2^j z_i]|}{2^j} \right)^{10/D^{1.2}} - \frac{10}{D^2} + f_{pen}(x)$$

$z = Q \wedge^{100} R(x - x^{opt})$



(a) f_1 unimodal sphere

(b) f_8 Rosenbrock

(c) f_{17} Schaffers F7

(d) f_{21} Gallagher's Gaussian 101-me Peaks

(e) f_{23} Katsuura

(f) f_{102} Sphere uniform noise

(g) f_{110} Rosenbrock Gaussian noise

(h) f_{124} Schaffers seldom Cauchy noise

(i) f_{128} Galagger's Gaussian noise

Figure 3: Plots of the bbob functions used in the test suite with D=2 [3]

3.1.2 Noisy Functions

Three noise models are used: uniform (UN), Gaussian (GN) and Cauchy (CN).

The uniform noise model multiplies the function with noise as shown in the equation below. We will set strength parameters α to $0.01(0.49 + 1/D)$ and β to 0.01, as recommended by Flink et al (2009)[4] for creating moderate noise. Because the noise strength depends on the value of f, optimal regions will be noisier. Compare, for example, figure 1 (a) with figure 1 (f).

$$f_{UN}(f, \alpha, \beta) = f \times \mathcal{U}(0,1)^{\beta} \max\left(1, \left(\frac{10^9}{f+\epsilon}\right)^{\alpha \mathcal{U}(0,1)}\right)$$

The Gaussian noise Model multiplies a function by log-normally distributed noise. We will set the noise strength parameter β to 1. See for example figure 1 (g) and figure 1 (i).

$$f_{GN}(f, \beta) = f \times \exp(\beta \mathcal{N}(0,1))$$

The Cauchy model adds noise only to a few part of the function values, resulting in occasional large outliers, like in figure 1 (h).

$$f_{CN}(f, \alpha, p) = f + \alpha \max\left(0, 1000 + \mathbb{1}_{\{\mathcal{U}(0,1)<p\}} \frac{\mathcal{N}(0,1)}{|\mathcal{N}(0,1)| + \epsilon}\right)$$

The selected noisy functions are the same as the multimodal functions, except that noise is applied to them.

f_{102} **Sphere with moderate uniform noise**

$$f_{102}(x) = f_{UN}\left(||z^2||, 0.01\left(0.49 + \frac{1}{D}\right), 0.01\right) + f_{pen}(x) + f_{opt}$$

f_{110} **Rosenbrock with Gaussian noise**

$$f_{rosenbrock}(x) = \sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2)$$

$$f_{110}(x) = f_{GN}(f_{rosenbrock}(x), 1) + f_{pen}(x) + f_{opt}$$

f_{124} **Schaffers F7 with seldom Cauchy noise**

$$f_{schaffer}(x) = \left(\frac{1}{D-1}\sum_{i=1}^{D-1}\sqrt{s_i} + \sqrt{s_i}\sin^2\left(50 s_i^{1/5}\right)\right)^2$$

$$f_{124}(x) = f_{CN}(f_{schaffer}(x), 1, 0.2) + f_{pen} + f_{opt}$$

f_{128} **Gallagher's Gaussian peaks, 101-me with Gaussian noise**

$$f_{gallagher}(x) = T_{osz}\left(10 - \max_{i=1}^{101} w_i \exp\left(-\frac{1}{2D}(x - y_1)^T R^T C_i R(x - y_i)\right)\right)^2$$

$$f_{128}(x) = f_{GN}(f_{gallagher}(x), 1) + f_{pen} + f_{opt}$$

3.2 Experimentation

The experiment was run in R using the 'mlrMBO' package. It creates the initial design, computes the Gaussian process surrogate model and and creates and maximizes the acquisition function. On every function in the test suite and for every dimension $d \in [2, 5, 10, 20, 40]$, the BO algorithm was run 30 times for 100 iterations. Every run was started with an initial design acquired by the Maximin LHS function, because we want these points to be spread out over the objective functions' domain as much as possible. The Matèrn covariance function with $\zeta = 3/2$ was used to construct the priors as this showed to have a preferable degree of smoothness in similar experiment [5]. The squared exponential function with $\kappa = 0.5$ was used as acquisition function to create a balance between exploration and exploitation. The results were scaled to their objective function's mean and optimum to make the results comparable over all the objective functions.

The noise was added to or multiplied by the objective function as described in the test suite section. Afterwards, the points at which the noisy functions were evaluated, were evaluated again on the function without the noise. This was done because the noise is meant to mimic measurement error, and if an evaluation is one good due to the measurement error, it should not be considered a good evaluation.

4 Results

Table 1 shows the average, standard deviation and the best found solution of both algorithms after 100 evaluations, of 30 runs per function and dimension. The results are scaled to $[0, 1]$ where 0 is the optimum of the objective function and 1 the mean, or the expected value of a random function evaluation. Averaged paths of progress during the runs are plotted in figure 4 per dimension. The BO results (blue) for $d = 20$ shows a dip at evaluation 21 and for $d = 40$ at evaluation 41 because the BO algorithm needs d observations to compute a prior. The progress made by evaluations before those is due to the Maximin LHS initial design. The CMA-ES results show a step-wise pattern because one iteration of the algorithm requires multiple evaluations.

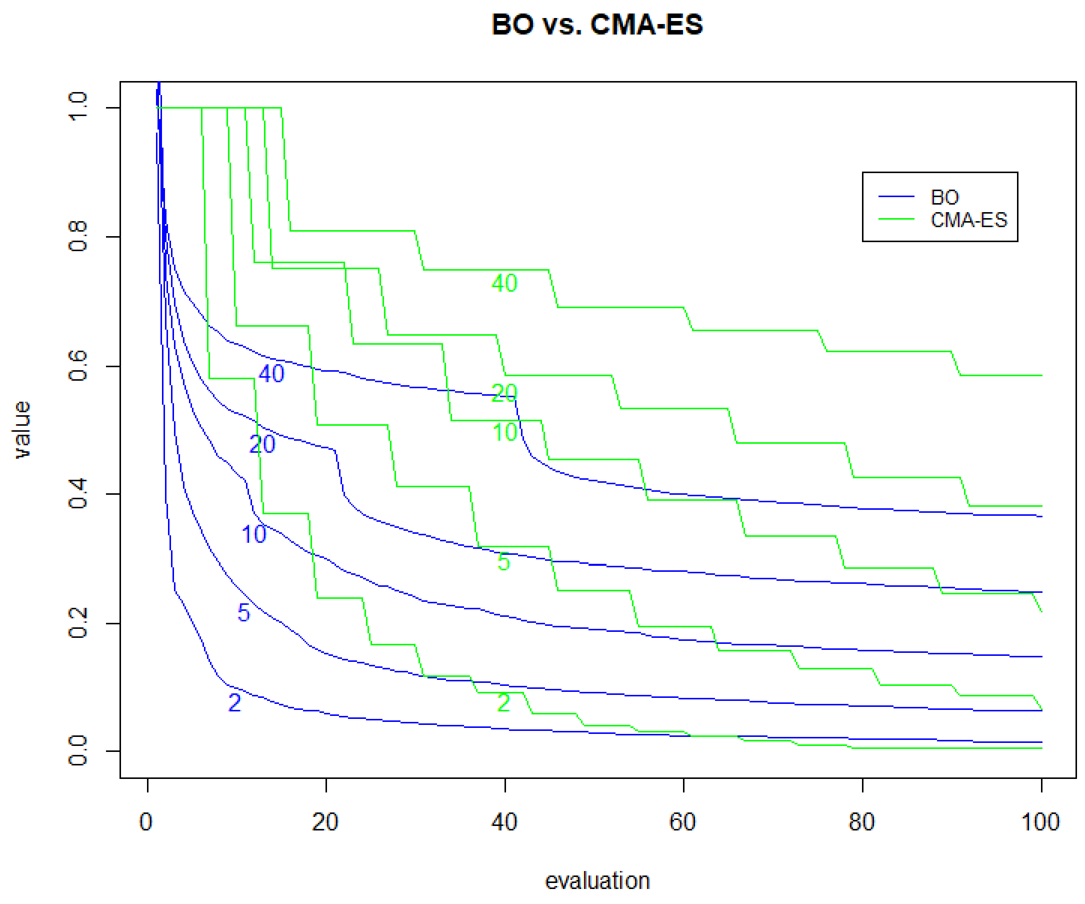


Figure 4: Average results on the test-suite for different dimensionalities, BO in blue, CMA-ES in green.

On average, the BO algorithm achieves a greater improvement in the first few evaluations than CMA-ES. Even in the initial design phase BO outperforms CMA-ES. It looks like CMA-ES might find better results beyond 100 iterations but we have no data on that. it also seems that these differences between the two strategies is amplified for greater dimensions.

In figure 5 averaged results are plotted for the different objective functions for 40 dimensions. While the BO seems to yield reasonable results for most objective functions, f21 and f128 find almost no improvement at all, not even by the initial design. These functions are characterized by not having any global structure. For Katsuura and Schaffer's with Cauchi noise BO was not able to make an informative prior that fit the objective function even remotely, causing it to get stuck in one area. CMA-ES also has more trouble with some functions than with others, but works on all of them, and keeps improving incrementally, even past 500 evaluations. It works especially well on Schaffers F7 with and without noise.

Figure 6 shows the effect of noise on the results after 100 iterations. The results on the noiseless functions (in green) are plotted alongside those of their noisy counterpart (in red). For BO t-tests

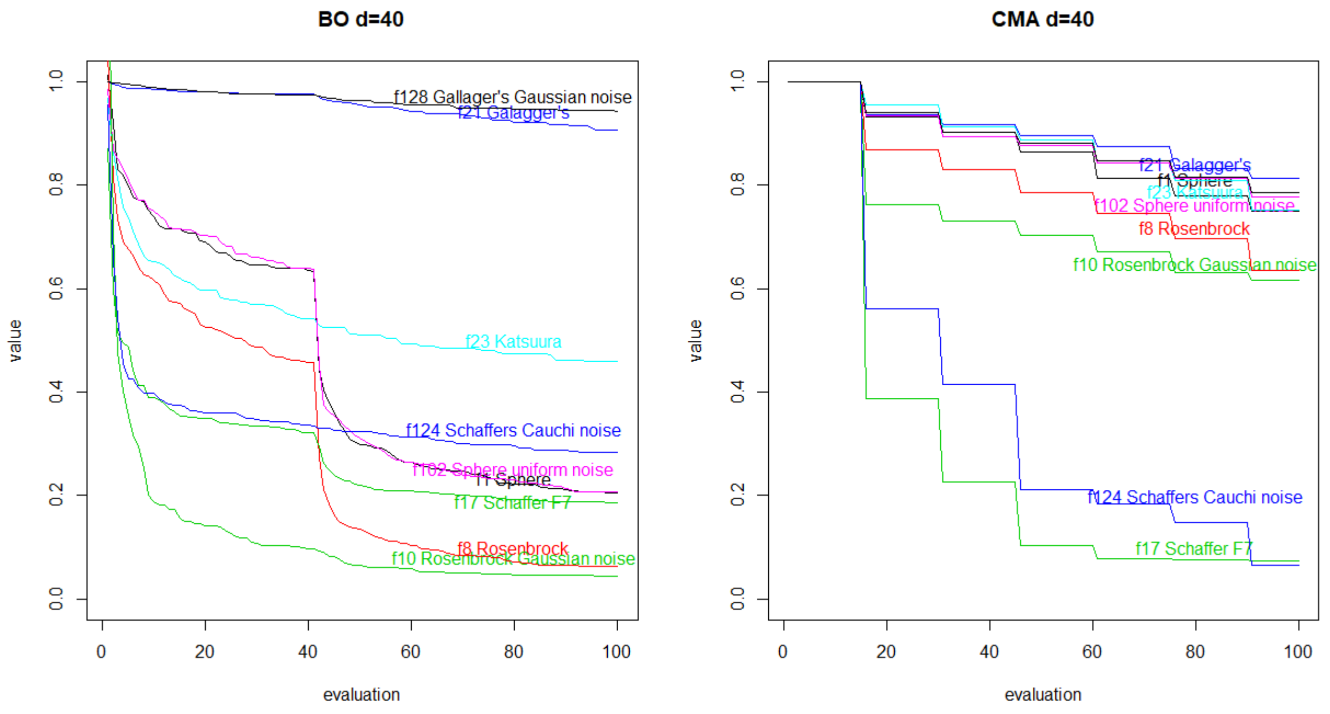


Figure 5: Averaged runs plotted per 40 dimensional objective function, left BO, right CMA-ES.

confirm that for four out of five functions, adding noise impacts the effectiveness of the algorithm negatively. For CMA-ES the effect of noise is not clear.

Figures 7 and 8 show the effect of the different types of noise on the progress during runs. The top left plots in both figures show the sphere function with and without uniform noise. The top right plots show the Rosenbock function with and without Gaussian noise. the bottom right plots show the Schaffer's F7 function with and without seldom Cauchi noise. The bottom right plots show the Gallaghers Gaussian 101-me Peaks function with and without Gaussian noise.

CMA-ES shows to be resilient to all types of noise tested in this research, as for each function CMA-ES had runs with some progress toward finding the optimum. For BO, noise mostly effects the results negatively, but could have no effect, or even be beneficial to the progress sometimes. Gaussian noise seems to be beneficial to the initial design on the Rosenbrock function, where the slope leading to the optimum changes direction D-1 times. Gauchi noise causes BO not to work properly on Schaffer's F7, which could be explained by the detrimental effect that an extreme outlier has on making the prior function.

A big difference between BO and CMA is in the running time of the algorithm. All the tests of CMA could have been run in in several seconds while BO took hours. Especially at 40 dimensions one iteration of BO took on average 2.21 minutes on a 3.4GHz cpu.

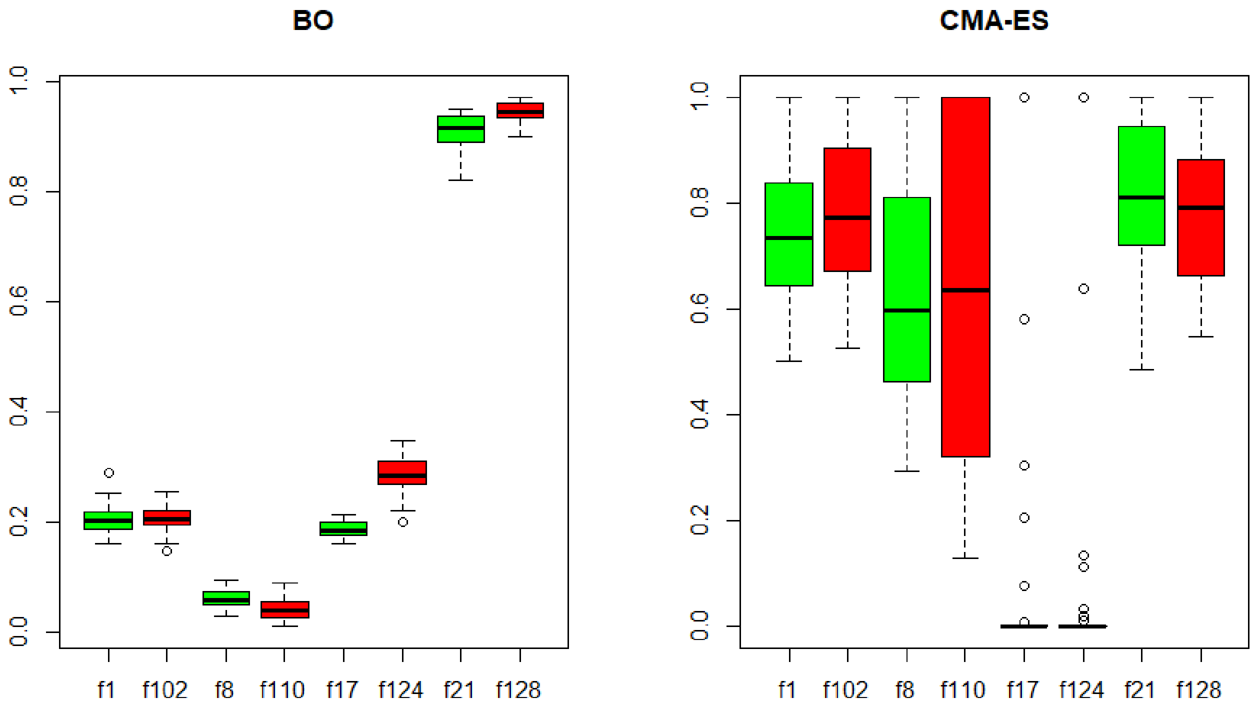


Figure 6: Boxplots comparing the noiseless functions (in green) with their noisy counterparts (in red). he values shown are the best values found after 100 iterations.

BO															
d	2			5			10			20			40		
	mean	sd	best	mean	sd	best	mean	sd	best	mean	sd	best	mean	sd	best
f1	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.001	0.000	0.052	0.012	0.029	0.205	0.028	0.162
f8	0.000	0.000	0.000	0.001	0.000	0.000	0.004	0.003	0.001	0.017	0.018	0.003	0.062	0.016	0.029
f17	0.001	0.001	0.000	0.035	0.015	0.012	0.031	0.007	0.012	0.057	0.010	0.036	0.186	0.016	0.161
f21	0.012	0.024	0.000	0.048	0.038	0.000	0.256	0.197	0.012	0.718	0.126	0.390	0.907	0.037	0.822
f23	0.080	0.045	0.016	0.182	0.058	0.069	0.248	0.059	0.128	0.338	0.064	0.229	0.458	0.072	0.234
f102	0.009	0.012	0.000	0.013	0.006	0.002	0.021	0.008	0.006	0.052	0.013	0.027	0.206	0.025	0.147
f110	-0.008	0.001	-0.009	0.003	0.004	-0.001	0.009	0.007	0.001	0.022	0.013	0.003	0.043	0.019	0.012
f124	0.006	0.003	0.001	0.088	0.032	0.015	0.071	0.019	0.044	0.119	0.024	0.088	0.284	0.037	0.201
f128	0.041	0.041	0.000	0.202	0.100	0.037	0.691	0.114	0.410	0.862	0.039	0.759	0.944	0.021	0.901
CMA-ES															
d	2			5			10			20			40		
	mean	sd	best	mean	sd	best	mean	sd	best	mean	sd	best	mean	sd	best
f1	0.001	0.001	0.000	0.048	0.040	0.006	0.186	0.093	0.051	0.519	0.187	0.226	0.749	0.137	0.500
f8	0.003	0.014	0.000	0.010	0.023	0.000	0.090	0.106	0.009	0.324	0.250	0.091	0.635	0.225	0.292
f17	0.001	0.008	0.000	0.138	0.237	0.000	0.226	0.319	0.000	0.005	0.019	0.000	0.073	0.213	0.000
f21	0.002	0.004	0.000	0.039	0.030	0.003	0.200	0.120	0.049	0.516	0.137	0.265	0.814	0.144	0.484
f23	0.033	0.170	0.000	0.045	0.045	0.009	0.263	0.182	0.061	0.537	0.222	0.113	0.753	0.153	0.504
f102	0.000	0.001	0.000	0.042	0.030	0.003	0.222	0.120	0.044	0.529	0.180	0.197	0.777	0.141	0.527
f110	0.001	0.003	0.000	0.015	0.042	0.000	0.113	0.198	0.002	0.470	0.291	0.114	0.617	0.321	0.130
f124	0.002	0.003	0.000	0.183	0.269	0.001	0.387	0.400	0.000	0.015	0.031	0.000	0.065	0.213	0.000
f128	0.000	0.000	0.000	0.087	0.114	0.002	0.276	0.242	0.051	0.513	0.176	0.225	0.785	0.140	0.549

Table 1: Mean, standard deviation and value of best found point after 100 evaluations. 0 represents the objection function's optimum and 1 the expected value of a random observation.

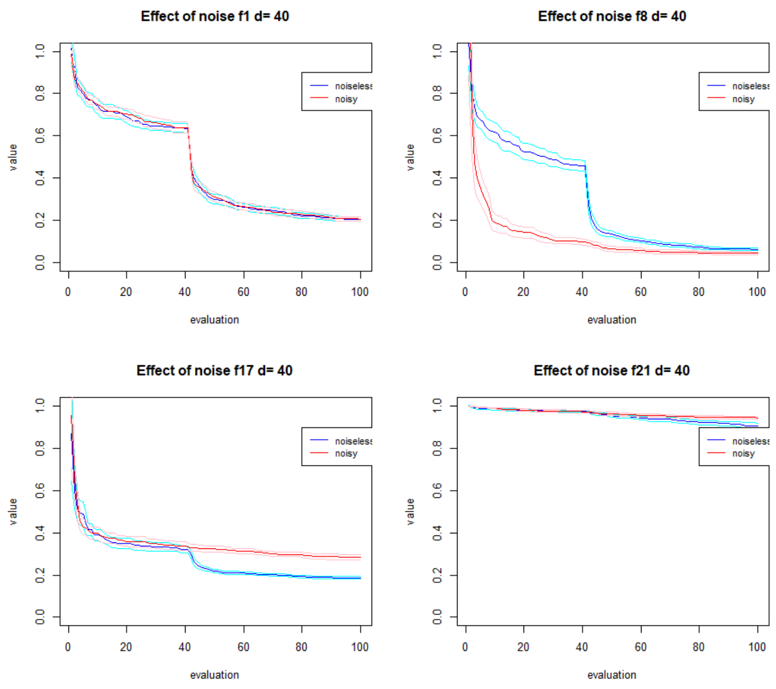


Figure 7: BO results comparing noiseless function to their noisy counterparts. a) sphere noiseless and with uniform noise; b)Rosenbrock noiseless and with Gaussian noise; c)Schaffers F7 noiseless and with seldom Cauchi noise; d) Gallager's noiseless and with Gaussian noise.

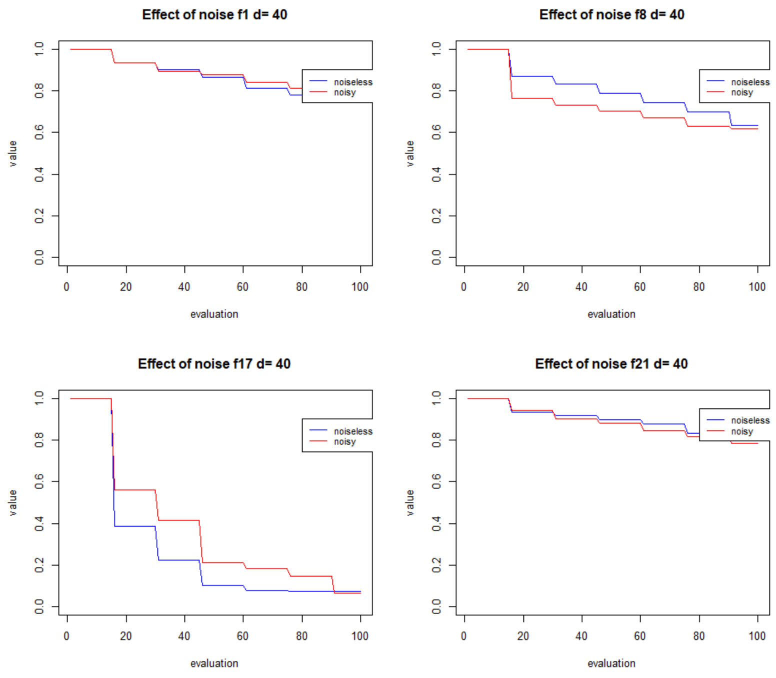


Figure 8: CMA-ES results comparing noiseless function to their noisy counterparts. a) sphere noiseless and with uniform noise; b)Rosenbrock noiseless and with Gaussian noise; c)Schaffers F7 noiseless and with seldom Cauchi noise; d) Gallager's noiseless and with Gaussian noise.

5 Discussion

We are led to the conclusion that that, on average, BO is more sample-efficient than CMA-ES. This implies that robots learning to locomote would need fewer trial runs to achieve some acceptable but sub-optimal level of movement. However, BO does not work well in all of the cases we researched, sometimes not making any progress at all. While CMA-ES is less sample-efficient, it is more robust, which would make it less likely that some robots will not be able to learn to locomote at all. We found CMA-ES to be resilient to different types of noise, which means that measurement errors would be less of a concern.

Furthermore, it is disputable whether the sample-efficiency of BO in this research can actually be ascribed to the BO algorithm working well, as the bulk of the early progress is made in the initial design phase, before the first surrogate model has even been constructed. Because BO requires d initial evaluations, it would seem unpractical to use this algorithm if the dimensionality is close to the amount of evaluations we can afford. Perhaps we should conclude that maximin LHS, our choice for the initial design of BO, outperforms CMA-ES in terms of sample-efficiency.

Also the running time of BO could be problematic for the application of evolutionairy robotics. If a robot would need to compute for two minutes between one-minute trial runs, an evaluation becomes three times as expensive and CMA-ES would be more time-efficient.

5.1 Combining methods

In the implication of robots learning to locomote, a combination of BO and CMA-ES could be appropriate. In the first d iterations we could use BO, or its initial design, as this is more sample-efficient than CMA-ES and computationally inexpensive. After d iterations it quickly becomes clear whether BO is effective for the particular morphology or objective function. If it appears that BO works well, most of the progress is made in, say, 5 or 10 following evaluations. Therefore, it would be advisable to use BO then for a couple of iterations, even if it takes several of minutes to compute. If BO appears not to work, CMA-ES would be a good alternative to continue with from this point as it seems to give some guarantee of incremental improvement. The best couple of points found so far would form the initial population.

5.2 Future work

A useful topic for future work on BO is reducing the computation time of creating the surrogate model by making an approximation. Lizotte (2008) [5] mentions two possibilities: subset of data Points approximation and projected process approximation.

Also the running time can be decreased using parallel runs, meaning that every iteration two points are suggested and evaluated. The amount of times that a surrogate model has to be computed would be halved, but the suggested points might be less optimal.

The running time could be less bothering if the computation is done while a robot is performing a trial run. It would cause a delay in the process of updating the surrogate as the evaluation of iteration t is added to the data in iteration $t + 2$, but there would be less time wasted waiting for the computations.

In the first d iterations BO is more efficient than CMA-ES only because of the initial design. Because we only tested maximin LHS in this research, improvement could possibly be made with a different algorithm for the first d iterations.

References

[1] V. M. Cora E. Brochu and N. de Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. 2010.

[2] N. Hansen. The cma evolution strategy: A comparing review. 2006.

[3] R. Ros S . Finck, N. Hansen and A.auger. Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions. *Research Center PPE*, Technical Report 2009/20, 2009.

[4] R. Ros S . Finck, N. Hansen and A.auger. Real-parameter black-box optimization benchmarking 2009: Presentation of the noisy functions. *Research Center PPE*, Technical Report 2009/21, 2009.

[5] D. J. Lizotte. Practical bayesian optimization. 2008.