EDISON

Electromagnetic Design of
flexIble SensOrs

FNP Team-Tech

Artificial intelligence and machine learning for computer aided design of microwave circuits and antennas

Report 2. Further testing of Bayesian Optimization packages.

Michał Baranowski
December 18, 2020

European Funds Smart Growth

FNP Foundation for Polish Science

European Union
European Regional Development Fund

The „EDISOn - Electromagnetic Design of flexIble SensOrs" project, agreement No. TEAM TECH/2016-1/6, is carried out within the TEAM-TECH programme of the Foundation for Polish Science co-financed by the European Union under the European Regional Development Fund.

1 Introduction

The first part of this report describes custom benchmark goal functions chosen for testing Bayesian Optimization algorithms for microwave & RF design problems. The second part of the report describes the tests of Bayesian Optimization (BO) packages which have been carried out so far. Future goals and further work plans are briefly discussed in the conclusion.

2 Goal functions for RF problems

This section contains a brief description of the goal functions that have been used for testing BO packages.

Multisection impedance transformer

In order to test various BO packages with a simple RF design problem, the N-section impedance transformer in microstrip technology has been chosen. The design procedure and suitable analytical equations have been well described in [1] and used to define the goal function. The goal is to minimise the overall reflection coefficient over a certain bandwidth using N microstrip line sections of length equal to $\lambda_0/4$. The impedance of the sections is controlled by changing their width.

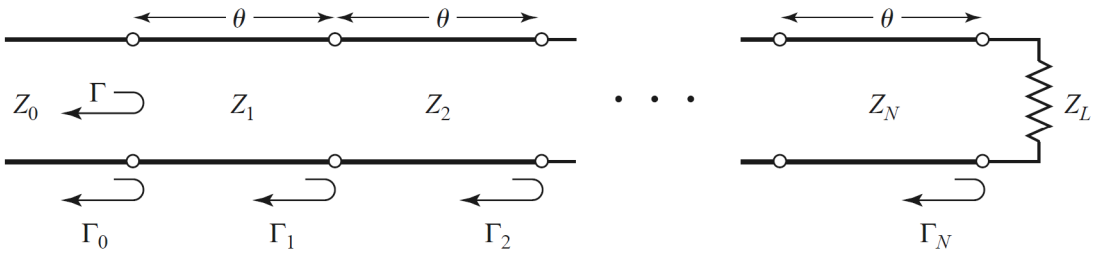


Figure 1: multisection matching transformer [1]

This problem is described by analytical equations only, thus it is very fast to calculate and can be easily rescaled to a high dimensional problem (by changing the number of sections, N). It has been written in Matlab, Python and C++, and can be utilized in each of the packages analyzed so far.

Waveguide cavity filters in InventSim

In order to test a real EM simulated RF problem, two waveguide cavity filter projects have been prepared in InventSim. The first one is controlled by four parameters, which stand for the widths of the irises and length of the cavities. The second one is controlled by five parameters describing the heights of the metallic posts. Both projects are shown in Fig. 2 below.

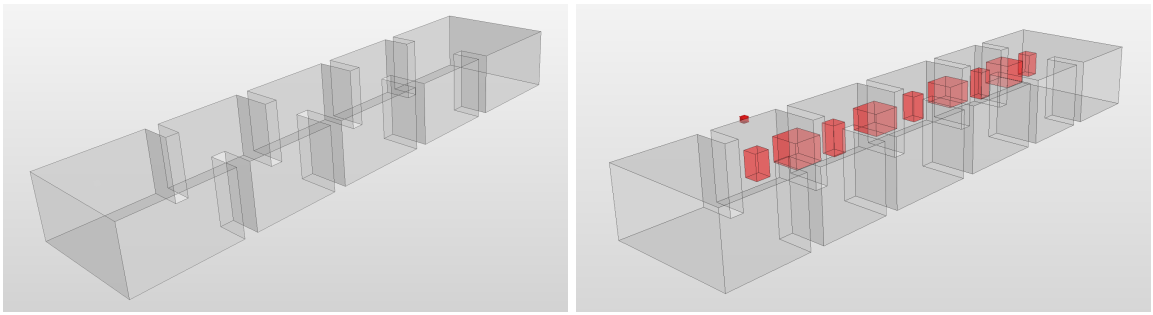


Figure 2: left: filter 1 (4 design variables), right: filter 2 (5 design variables)

The goal here is to find a desired filter response in a given passband. Each function evaluation requires to run a EM simulation to calculate the S11 and S21 parameters. The goal is calculated as a

root mean square of the errors at each frequency point in the selected passband.
This project and goal function have been used in Matlab and Python thanks to the use of COM interface, which allows to access InventSim variables and simulation directly from the code.

Waveguide filter with mode-matching method

The third type of goal function used to test some of the packages is a waveguide filter with metallic plates inside, as shown in Fig. 3.

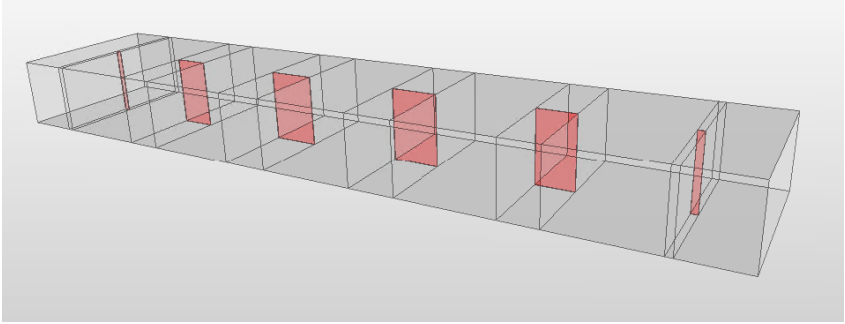


Figure 3: waveguide filter with metallic plates

The goal is to choose optimal values for the subsequent waveguide sections to design a filter implementing the Chebyshev band-pass filtering function. The default number of parameters for this problem is 6, but it can be easily rescaled for more.
The codes have been developed by prof. Adam Lamecki and are written in Matlab, but can be rewritten to Python or C++ if necessary.

3 Bayesian Optimization packages

3.1 Software setup

The workspace for Bayesian Optimization testing has been set up on Python 3.6 virtual environment under Windows 10 operating system. The **bayeso** and **Dragonfly** packages, as well as suitable libraries enabling access to the InventSim solver through COM interface have been installed.
The **Cornell-MOE** and **limbo** packages were installed on the Ubuntu system under Windows Subsystem for Linux (WSL).The **DPT-BO** package works on Matlab under any operating system.

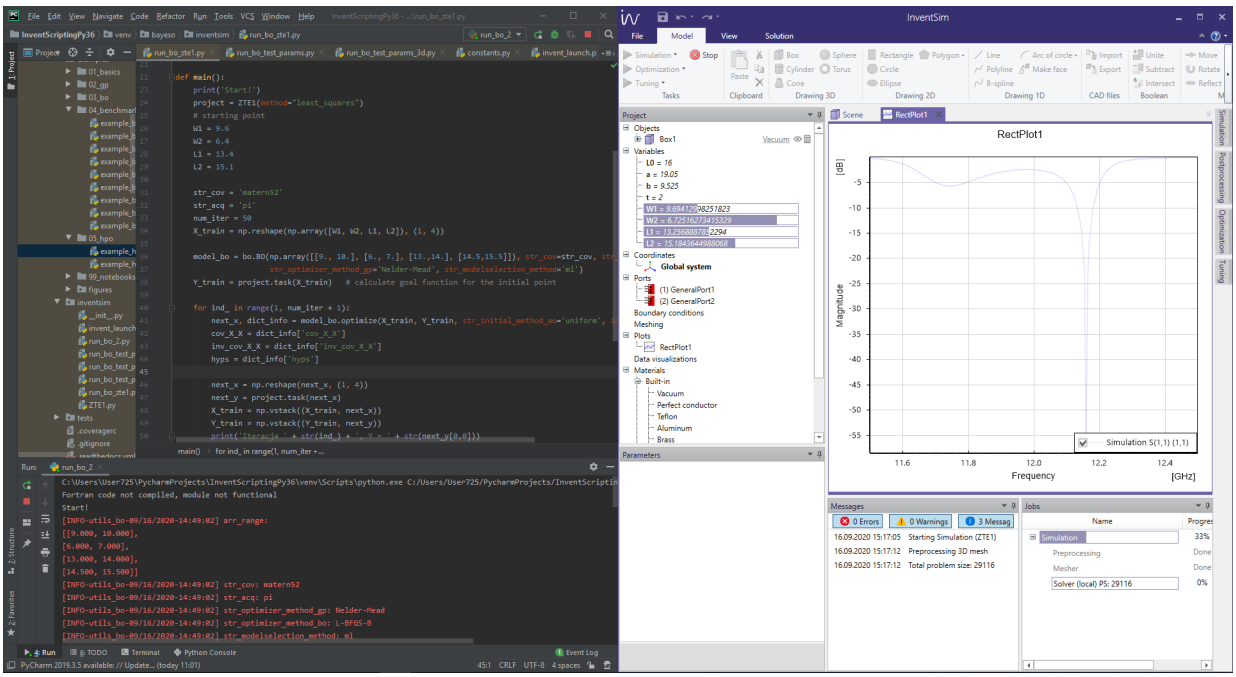
3.2 bayeso package

The first package chosen to carry out tests on was bayeso [2]. It is relatively simple and the code is well documented, therefore it was chosen as a good starting point for learning to use Bayesian Optimization. The package offers three kernel covariance functions: Squared Exponential (Exponentiated Quadratic), Matern 3/2 and Matern 5/2, as well as six acquisition functions: Augmented Expected Improvement (AEI), Expected Improvement (EI), Probability Improvement (PI), pure exploitation, pure exploration, and Upper Confidence Bound (UCB).

The optimization efficiency for high-dimensional problems has been tested on several benchmark functions implemented in the package, including Branin (2-dim), Ackley (N-dim), Rosenbrock (N-dim), Hartmann (6-dim) and others. Although the optimization with default settings does well for one-, two- and three-dimensional problems, it fails to minimize some synthetic functions of dimension 4 and higher. The benchmarking results are summarized in the table below.

Function	No. of dimensions	Optimization result
Branin	2	optimized (20 function evaluations)
Ackley	2	optimized (100 function evaluations)
Rosenbrock	2	optimized (100 function evaluations)
Ackley	3	optimized (100 function evaluations)
Rosenbrock	3	optimized (100 function evaluations)
Hartmann	6	optimized (100 function evaluations)
Ackley	4	not optimized (400 function evaluations)
Rosenbrock	4	not optimized (200 function evaluations)

To test Bayesian Optimization for CAD of microwave components, the goal function with a waveguide cavity filter optimization (4 design variables) has been chosen.
In order to return a scalar, the output values (calculated for each frequency point) are represented as a sum of squares. The same scalar goal function has been used with Matlab *fmincon* optimization, and has shown to converge quickly: 65 function evaluations (good starting point), 100 function evaluations (bad starting point).



The conducted tests have shown that bayeso optimizes the filter design with two and three project variables, but the problems arise with four project variables - then the algorithm requires ca. 200 iterations to find the satisfying solution (for good starting point or random initialising with 5 points). The 4D goal function has been tested for every combination of available acquisition and covariance functions. Although the results are not always satisfying, the tests have shown that the best covariance function is Matern 5/2, while the best acquisition function is Probability Improvement (but it is not the best choice in every situation, however).

After encountering troubles with the first custom goal function, the bayeso package was no longer investigated. It could be later tested with the impedance transformer optimization problem to have a better comparison with other packages and methods.

3.3 Dragonfly package

The second package chosen for BO testing was Dragonfly [3]. It is a Python library for scalable BO, especially suited for high dimensional optimization problems. The package was tested with three custom goal functions: both InventSim filter optimization problems (with 4 and 5 design variables)

and the multisection transformer (with 6, 8 and 10 parameters). The results are summarized in the table below.

Function	No. of dimensions	Optimization result
filter 1	4	not optimized (100 function evaluations)
filter 1	4	optimized (250 function evaluations)
filter 2	5	not optimized (250 function evaluations)
transformer	6	optimized (300 function evaluations)
transformer	8	not optimized (400 function evaluations)
transformer	10	not optimized (400 function evaluations)

The package optimizes the four dimensional filter design, but fails to optimise the five dimensional project. It does find the minimum for 6D transformer goal function but fails to get the desired minimum for higher dimensions. For comparison, the same transformer goal function was optimized by Matlab *fmincon* in less than 200 function evaluations, with 8 and 10 dimensional input. More details about the performed tests and optimization results in Dragonfly can be found at dragonfly results spreadsheet.

3.4 DPT-BO package

Another package under testing was DPT-BO: High dimensional Bayesian Optimization with Deep Partitioning Tree. This method is the only package that has been presented in use for high-frequency electronic design problems. The code is written in Matlab and thus can be easily coupled with external optimizers and software, such as InventSim EM simulator. The package was tested with all goal funtions developed for this purpose: both InventSim waveguide filter projects, the waveguide filter computed with the mode-matching method and the N-section impedance transformer. The results are shown below.

Function	No. of dimensions	Optimization result
filter 1 (minmax)*	4	not optimized (150 function evaluations)
filter 1	4	not optimized (250 function evaluations)
filter 2	5	not optimized (100 function evaluations)
modematch	6	not optimized (400 function evaluations)
transformer (minmax)	6	not optimized (250 function evaluations)
transformer	6	optimized (250 function evaluations)
transformer	8	optimized (250 function evaluations)
transformer	10	not optimized (400 function evaluations)

* the type of output scalar goal function: minmax or RMSE (root mean squared error). If not stated otherwise, the RMSE output was used.

As can be observed, the DPT-BO package copes quite well with the transformer example, but fails with every microwave filter goal function, regardless of the input dimension and method of computation. It is a disappointing result, taking into consideration that the DPT-BO method was told to be developed especially for solving microwave & RF design problems. More about the results can be found at DPT-BO results spreadsheet.

3.5 Cornell-MOE package

The Cornell-MOE package [5] has been chosen for testing because of various add-ons and improvements for high-dimensional problem optimization as well as the possibility to use gradient information during the optimization process. Unfortunately, the package is no longer supported and causes numerous problems during setup. Some of the problems were overcome, which enabled the use of the package standard methods, Expected Improvement and Knowledge Gradient, but the gradient-enabled methods are still not available for the user (any attempts to use them throw some C++ type conversion related

problems).
For that reason, only the standard EI and KG method were tested and the results were unsatisfying, so further work on this package has been ceased. The results for some benchmark and custom goal functions are listed below.

Function	No. of dimensions	Optimization result
Branin (EI)	2	optimized (50 function evaluations)
Hartmann (EI)	3	optimized (50 function evaluations)
Hartmann (EI)	6	not optimized (100 function evaluations)
Hartmann (KG)	6	optimized (100 function evaluations)
transformer	6	not optimized (250 function evaluations)
transformer	8	not optimized (250 function evaluations)

More about these results can be observed at cornell-moe results spreadsheet.

3.6 limbo package

The last of the investigated packages is limbo [6]. The package is written in C++, can be run on Linux OS and is aimed for performing BO in the area of automation and robotics. The package has been tested with synthetic benchmark function Hartmann 6D as well as the impedance transformer function (with 6,8,10 and 12 input variables). The results are listed in the table below.

Function	No. of dimensions	Optimization result
Hartmann	6	optimized (200 function evaluations)
transformer	6	optimized (200 function evaluations)
transformer	8	optimized (400 function evaluations)
transformer	10	optimized (400 function evaluations)
transformer	12	not optimized (600 function evaluations)

The results mentioned above are all acquired with limbo running without GP hyperparameter optimization during the optimization process (in comparison with e.g. DPT-BO, where the algorithm optimizes the GP model in every iteration, or Dragonfly, where best results where obtained with GP hyperparameter optimization in every 3 iterations). First tests of limbo hyperparameter optimization showed worse results than without, but this package requires further research and testing. The results can be found at limbo results spreadsheet.

4 Conclusions

The conducted tests have shown that Bayesian Optimization encounters problems with high-dimensional function optimization, particularly with RF filter design problems. Each package has their own hyperparameters / options to set, which affect the optimization process or how the GP model is being built. The packages mentioned in this report were tested mostly with just 2-3 custom goal functions and definitely can be further investigated. However, the results obtained so far give an overall conclusion about the usefulness of the BO algorithms in computer aided design of microwave & RF components, and these results show that this approach may not be very effective in this area of science.

5 Further work

The next steps may be focused on further testing of the aforementioned packages, especially limbo, which has not been investigated thoroughly yet. Some time may be needed to overcome the software limitations (e.g., being able to run COM interface commands directly from C++, running InventSim software from WSL Ubuntu).
Also, to be able to compare these packages, a unified set of tests should be run on every package

with possibly the same options enabled. At the moment, each package has been tested with slightly different test functions and/or options (number of function evaluations, GP optimization and other options differ).
In order to continue searching for the fast and efficient BO-related optimization algorithm, some more methods and packages may be tried out. Among the most promising methods left there is the **dlib** library with MaxLIPO+TR algorithm based on a method proposed by [7]. It is advertised as follows:

'[Trying to use] Bayesian Optimization: Use a tool like MATLAB's bayesopt to automatically pick the best parameters, then find out Bayesian Optimization has more hyperparameters than your machine learning algorithm, get frustrated, and go back to using guess and check or grid search.
Instead, try the MaxLIPO+TR. What is remarkable about this method is its simplicity. In particular, MaxLIPO+TR doesn't have any hyperparameters, making it very easy to use.'

More about this algorithm can be found on dlib blog page here.

References

[1] D. Pozar, "Microwave Engineering", John Wiley & Sons, 2009.

[2] J. Kim, S. Choi, "bayeso: A Bayesian optimization framework in Python", `http://bayeso.org`, 2017.

[3] K. Kandasamy et al., "Tuning Hyperparameters without Grad Students: Scalable and Robust Bayesian Optimisation with Dragonfly", Journal of Machine Learning Research, 2020. package: `https://github.com/dragonfly/dragonfly/`

[4] H. Torun, M. Swaminathan, "High dimensional Bayesian Optimization with Deep Partitioning Tree", `https://github.com/hakkimerttorun/DPTBO`.

[5] J. Wu et al., "Cornell-Metrics-Optimization-Engine", `https://github.com/wujian16/Cornell-MOE`.

[6] A. Cully, K. Chatzilygeroudis, F. Allocati, J. Mouret, "Limbo: A Flexible High-performance Library for Gaussian Processes modeling and Data-Efficient Optimization", `https://github.com/resibots/limbo`.

[7] C. Malherbe, N. Vayatis, "Global optimization of Lipschitz functions", Proceedings of the 34th International Conference on Machine Learning, Vol. 70, 2017.