Bridging Theory and Practice Through Modular Graphical User Interfaces

Frederik Rehbach
TH Köln
Institute for Data Science, Engineering and Analytics
Steinmüllerallee 1
51643 Gummersbach, Germany
frederik.rehbach@th-koeln.de

Jörg Stork
TH Köln
Institute for Data Science, Engineering and Analytics
Steinmüllerallee 1
51643 Gummersbach, Germany
joerg.stork@th-koeln.de

Thomas Bartz-Beielstein
TH Köln
Institute for Data Science, Engineering and Analytics
Steinmüllerallee 1
51643 Gummersbach, Germany
thomas.bartz-beielstein@th-koeln.de

ABSTRACT

State-of-the-art evolutionary algorithms and related search heuristics are well suited to solve problems from industry. Unfortunately, easy to use graphical user interfaces (GUI) are not available for many algorithms. We claim that the availability of well-designed GUIs might increase the acceptance of these algorithms in the real-world domain. The spotGUI R-package, which is introduced in this paper, provides a GUI for the already well-established SPOT package. It includes state-of-the-art algorithms and modeling techniques that can be used without the requirement of optimization or programming knowledge. Using the spotGUI in industry, as well as education, delivered first promising results.

Keywords

SPOT, Graphical User Interface, Real-World Applications

1. INTRODUCTION

Industrial problems are highly complex and challenging for even the most advanced state-of-the-art algorithms. However, the difficulty in solving such problems is often not their high complexity, but rather the challenge for a non-expert user to apply a suitable algorithm. For a significant subset of the existing optimization problems in industry, suitable state-of-the-art algorithms already exist. Yet, they are often still not applied because they are

a) not known to the field specialist or
b) no simple implementation is available.

This paper presents a simple to use GUI that bridges the gap between existing algorithms and real-world problems. The core of the new package relies on the Sequential Parameter Optimization Toolbox (SPOT) [1]. SPOT provides a modular structure for combining sampling methods, modeling techniques and optimizers for an all-in-one Surrogate Model-Based Optimization (SMBO) toolbox. In SMBO, a data-driven surrogate model is fitted to the data of an expensive to evaluate objective function, e.g., a complex simulation or a real-world experiment. Under the assumption that the surrogate is cheap to evaluate, an extensive search on the model becomes feasible. The predicted candidate solution, which best fulfills some user-specified infill criterion (e.g. best model function value) is evaluated on the expensive objective function and further used to update the model. The process is repeated in an iterative fashion. A more in-depth explanation of SMBO and its applications can be found in [5] and [2].

SPOT has been further improved and developed for many years. Today the package provides a vast set of different models, optimizers, and sampling schemes, each of which can be configured to user specific requirements. The system was initially targeted to parameter optimization tasks, but is well suited to any costly to evaluate optimization problem. The availability of these methods together with their respective documentation in the R-package is a first step towards an easy to use modular optimization tool. However, SPOT remains a high-level toolbox, which requires user experience and some R programming skills. Furthermore, since R is rarely used by engineers in industry, this again leads to problems (a) and (b) as previously discussed. The presented spotGUI tries to address these problems by making the tools included in SPOT accessible to everyone through an easy to use graphical interface.

The rest of this paper is structured as follows: Section 2 gives an overview of the basic functionality and some conceptual ideas of the spotGUI. In Section3 two practical example applications for the spotGUI applied in industry are presented. One of which is the Electrostatic Precipitator (ESP) Problem, a current, costly-to-evaluate, discrete optimization problem from industry. Lastly, the software, future opportunities, and room for improvements are discussed in Section 4.

2. WORKFLOW

2.1 Availability

The spotGUI package shall give more users easy access to SPOT. All stable versions are available on CRAN. Development versions are published on GitHub. One of the primary goals of the spotGUI is to allow non-R-users and even non-programmers to use SPOTs model-based optimization techniques. Additionally, it can benefit experienced SPOT users by enabling a faster setup and even code generation which will be covered in more detail in Section 2.5. The spotGUI is developed in the R extension Shiny [4]. It is divided into

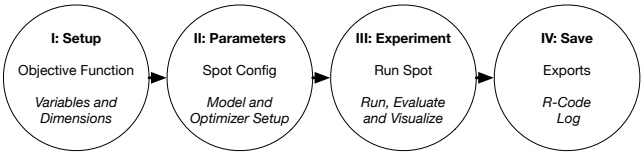

Figure 1: Typical optimization workflow for SMBO in the spotGUI

four separate tabs, arranged in a typical workflow order as presented in Figure 1 and Algorithm 2.1. Each of the tabs is explained in more detail in the following.

Algorithm 2.1: Surrogate Model-based Optimization

```
step I: setup
select and parametrize objective function
begin
    step II: parameters
    select and parametrize surrogate model
    select and parametrize experimental design
    step III: experiment
    generate design points
    evaluate design points with objective function
    build initial surrogate model
    while not termination-condition do
        search for optimum on surrogate model
        evaluate new point on the objective function
        update surrogate model
    end
    step IV: save
end
```

2.2 Setup

The objective function is specified and parametrized on the first setup tab. A screenshot of the configuration window is shown in Figure 2. Additionally to having an option to insert any function through the R-Environment and supporting manual result input, the spotGUI provides a broad set of preconfigured test functions.

The set of provided test functions is loaded from the 'smoof' R-package [3], which provides an interface to many single- and also multi-objective test functions. Of these, the spotGUI only includes the current set of single-objective functions, totaling in 76 test functions. Each of these functions is loaded with its respective bounds as well as dimensionality. Scalable functions are loaded as 2-dimensional functions and can then be adapted by the user to any desired dimensionality. The 'smoof' package also allows the user to filter the functions by specific tags such as "separable", "differentiable" or "weak-global-structure". This makes it possible to test a given optimizer on a particular type of test function that should behave somewhat similar to a real-world problem that shall be solved. Different settings for SPOT and its tools can quickly be tested by using the spotGUI with the given set of test functions.

The possibility to manually input evaluation results enables non-programmers to use the spotGUI without any requirements for an objective function definition in code. Thus for example making it possible to use SPOT to optimize

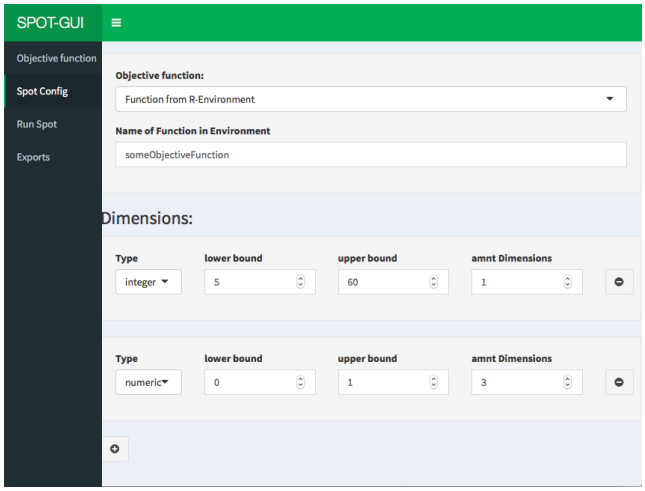

Figure 2: Screenshot illustrating the objective function setup in the spotGUI. The user has to define the function as well as it's dimensionality and variable types.

some real-world experiments by entering / importing the experiment results back into the spotGUI. The only configuration required in this scenario is to insert information on the problem dimensions. Each dimension is configured with a type (numeric/integer/factorial), as well as upper and lower bounds. If there are multiple dimensions with the same upper and lower bounds, the convenience option "amntDimensions" can be used to specify that the same bounds are required multiple times.

2.3 Parameters

One of the main benefits of the spotGUI becomes evident during the setup of SPOT itself. As previously mentioned SPOT features a wide variety of different models and optimizers, each of which again provides a variety of configuration options. In the spotGUI, these are conveniently selectable from drop-down menus. Showing each available option together with simple explanations through tooltips, tackles the requirement of any documentation reading for the user. The settings are arranged in four categories covering a general setup, modeling setup, optimizer setup and lastly design setup. Skipping the 'Spot Config' tab altogether results in a robust default setup for SPOT.

2.4 Experiment

The previously configured processes are executed in the "Run Spot" tab. The available options include creating a DOE, fitting a model, running a model-based optimization, and more. In the following, these methodologies will be briefly explained. In many expensive real-world applications, an initial screening for variable importance and interaction is desired. The spotGUI provides the option to do so with a configured sampling method to build a design of experiments. Depending on the objective function configuration, the generated experiments can be evaluated automatically or manually, e.g. a real-world experiment. Such manual results can either be imported into the spotGUI or directly entered into the result table. A surrogate-model is fitted to the given data making interactive 3D-visualizations available.

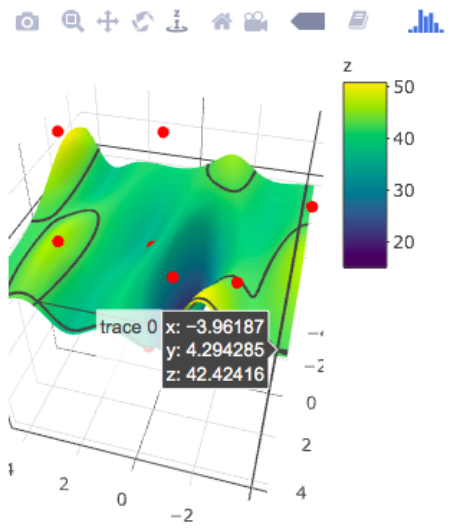

Figure 3: Auto generated plot showing the fitted surrogate model. Red dots indicate evaluated candidate solutions. Hovering the mouse over the plot results in the black info box showing more detailed information for the given plot location. The toolbar above the plots provides features for easy plot exports.

The graphics are generated through plotly, an R-library for creating web-based graphs [7]. The availability of interactive 3D plots enables the user to learn more about the landscape of their objective function intuitively and gives a deeper insight into variable behavior. After a model fit, it is easily possible to run an optimizer on the model to propose a single next candidate solution, thus enabling SMBO even to a manual user / non-programmer.

Further options are again aimed at enhancing the automatic evaluation and optimization of a configured objective function. As sometimes even just a few objective function evaluations might take a long time, the spotGUI execution can be interrupted and restarted from the last completed function evaluation. For users who only want to use the spotGUI as a quick setup tool for their code, another option exists. By entering the 'Log Only' mode, all computations that would usually be applied to the objective function are skipped. Instead, the actions are only written to the code log. From there they can be exported and used in any R-Script, enabling an extra fast setup for new SPOT projects.

2.5 Save

Each action that is executed in the spotGUI is written into an exportable R-Code log. The log is accessible on the 'Export' tab of the GUI, it can easily be exported or copied to the clipboard through the provided button. The resulting R-Code can be run standalone (given the spotGUI library is installed) and generates the same results as previously shown in the experimentation tab. This also ensures reproducibility of any work that was done with the help of spotGUI.

3. EXAMPLE APPLICATIONS

3.1 Applying the Manual Mode

The spotGUI offers a couple of functionalities to be easily usable and applicable to problems where real-world experiments are required. We can imagine the following example where the user is not too affine with software programming: A machine engineer who needs to set up a new metal hardening machine to deliver good performance.

Through the machine's interface, he is allowed to control two temperature parameters which define a temperature curve that the machine runs through in the hardening process. Additionally, he can change two time parameters which define the duration of the heating as well as the cooling phase in the hardening process. He is looking for the set of optimized parameters which result in the hardest end product. However, each test requires to run the hardening machine for a few hours and involves material costs. In this scenario, the *manual mode* of the spotGUI could help the engineer in this parameter optimization problem. First of all, by using the spotGUI in the manual mode, no coded fitness function is used. Instead, parameter settings are proposed by SPOT, manually evaluated on the hardening machine and inserted into the results table by the engineer.

The detailed workflow is as follows: After an initial setup in the spotGUI, defining the bounds and types of the input parameters, a DOE (Design of Experiments) is built. This is quickly done via the 'createDOE' button in the 'runMode' tab. A model can be fitted, and a visualization of it is available. With the now to him available information, the engineer could continue in a few different ways. He could straightforward accept the best solution found in the DOE. However, this should not be done if resources for more machine tests exist. Continuing with a more in-depth DOE, he could increase the DOE budget and optionally shrink the parameter bounds to an area that is considered as promising by the fitted model. The second option to spend the remaining test budget is to run an optimizer on the fitted model via the 'propose new point' functionality. This additional point is the model optimum for some configured infill-criterion. This criterion might be the best-predicted point, but depending on the model, it could for example also be the point with the highest expected improvement as utilized in EGO [6]. After evaluating the proposed point on the machine, the model can be refitted to include the new data point. After that, the 'propose new point' functionality is usable again. Therefore, by using this feature, surrogate model-based optimization is available in a manual use case, making a well-known and powerful optimization technique available to a broader audience. Lastly, the configuration of the spotGUI can easily be changed during the optimization process, allowing for a more interactive optimization approach.

3.2 The Electrostatic Precipitator Problem

Electrostatic precipitators (ESP)s are large scale electrical filtering/separation devices. They are used to remove solid particles from gas streams, such as from the exhaust gases of coal-burning power plants. An overview of the structure of an ESP can be seen in Figure 4. The illustrated separator has three central separation zones in which the

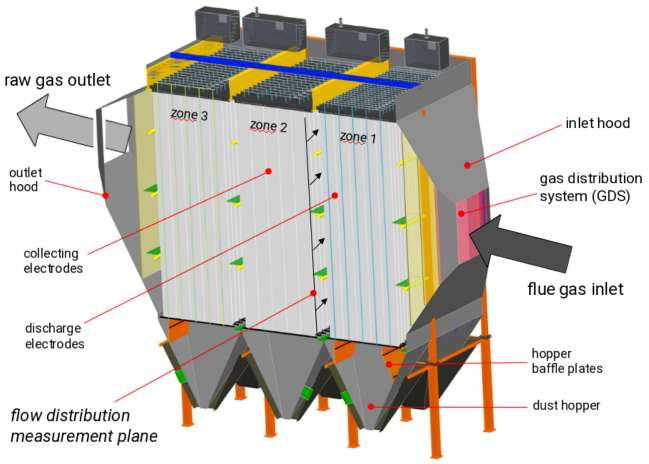

Figure 4: Electrostatic precipitator with 3 separation zones. This figure was kindly provided by Steinmüller Babcock Environment GmbH.

particles are separated from the gas flow by the precipitator. Gas streams in from piping through the inlet hood and exits through an outlet hood. The entrance and exit piping of the separator has a much smaller cross-section and therefore a higher gas velocity than desired in the separator. Without additional measures the fast gas stream would rush through the center of the precipitator, resulting in very low separation efficiency. The primary optimization target is the so-called gas distribution system (GDS). The GDS is mounted directly behind the flue gas inlet of the precipitator. It is used to distribute the gas flow from the small inlet cross-section to the much larger cross-section of the precipitation zones. The GDS in the given application consists of 49 configurable slots. Each of these slots can be filled with different types of metal plates, porous plates, angled plates, or be left completely empty. Increasing the separators efficiency by achieving a more evenly distributed gas flow allows a smaller overall separator. A reduced separator size, together with lowered operating costs would accumulate to multiple millions of euro in cost reduction.

Two central factors reveal a complex to solve optimization problem:

a) The amount of configurable slots together with the amount of available configurations per slot leads to $\approx 10^{41}$ possible configurations for the overall system
b) Each objective function evaluation requires a costly CFD-simulation in order to judge the gas flow through the system

The ESP optimization was approached with a combination of a parallelized model-based evolutionary algorithm that was equipped with newly created task-specific mutation and recombination operators. Tuning these operators was required in order to be able to reduce the overall runtime of each optimization to fit into standard project run times. In this industry project, the spotGUI was successfully applied to set up parameter tuning for the evolutionary algorithm and its operators.

4. SUMMARY

The SPOT package has been available for many years. It has been continuously updated and grew to a very large and useful platform. However, through the growing amount of possible configurations and use cases it simultaneously became more complex to dig through all settings and find the best ones for each problem. The here introduced spotGUI package reduces the configuration complexity back down to a level where any beginner can use the package. It was successfully applied to industry use cases as well as in student courses. Thus, demonstrating its ease of use and capability to provide easy to access visual information. The playful style with which different optimization methods can be applied makes the software a useful tool in education.

One of the most significant drawbacks of the current version of the spotGUI is its dependency on R. Till now, the spotGUI can only be published as a web application available through a browser or started directly in R. Future work on the spotGUI will, therefore, concentrate on making the software available as a standalone executable without the requirement of starting it through R. Additionally, more features are planned or even already are under construction, including: Parallelization support for SPOT, more DOE and analysis functionality, additional exports, and report generation.

5. ACKNOWLEDGEMENT

Parts of this work were supported by the "Ministerium für Kultur und Wissenschaft des Landes Nordrhein-Westfalen" (FKZ: 005-1703-0011).

6. REFERENCES

[1] T. Bartz-Beielstein and M. Zaefferer. A Gentle Introduction to Sequential Parameter Optimization. CIplus 1/2012, Fakultät 10 / Institut für Informatik, 2012.

[2] T. Bartz-Beielstein and M. Zaefferer. Model-based methods for continuous and discrete global optimization. *Applied Soft Computing*, 55:154 – 167, 2017.

[3] J. Bossek. smoof: Single- and Multi-Objective Optimization Test Functions. *The R Journal*, 9(1):103–113, 2017.

[4] W. Chang, J. Cheng, J. Allaire, Y. Xie, and J. McPherson. *shiny: Web Application Framework for R*, 2018. R package version 1.1.0.

[5] A. Forrester, A. Keane, et al. *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons, 2008.

[6] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

[7] C. Sievert, C. Parmer, T. Hocking, S. Chamberlain, K. Ram, M. Corvellec, and P. Despouy. *plotly: Create Interactive Web Graphics via 'plotly.js'*, 2017. R package version 4.7.1.