SIMBO - A FRAMEWORK FOR SIMULATION-BASED OPTIMIZATION USING BAYESIAN OPTIMIZATION

Philipp Zmijewski
Nicolas Meseth
University of Applied Sciences Osnabrück
Oldenburger Landstraße 62, Osnabrück 49090, Germany
Email: philipp.zmijewski@hs-osnabrueck.de, n.meseth@hs-osnabrueck.de

KEYWORDS

Simulation-based Optimization, Bayesian Optimization

ABSTRACT

SimBO is a flexible framework for optimizing discrete event-driven simulations (DES) using sequential optimization algorithms. While specifically designed for Bayesian Optimization (BO) in the context of DES, SimBO can be applied to any black-box problem with other optimization algorithms. The framework consists of four encapsulated components - the black-box problem, the sequential optimization algorithm, a database for experiment configuration and results, and a web-based graphical user interface - that communicate via well-defined interfaces. Each component can be run in different environments, allowing for cooperation between different hardware- and software configurations. In our research context, SimBO's architecture enabled BO algorithms to be run on a high-performance cluster with GPU support, while the simulation is executed on a local Windows machine using the Simio simulation software. The framework's flexibility also makes it suitable for evolving from a research-focused tool to a production-ready, cloud-based optimization tool for modern algorithms.

INTRODUCTION

Discrete Event Simulation (DES) is a commonly used approach for analyzing complex, dynamic, and stochastic systems. When enhanced with an optimization component, it becomes a powerful decision-making tool for such systems (Fowler et al. 2022). The approach of optimizing a system through a digital simulation is known as simulation-based optimization (SBO).

According to Riley (2013), simulation-based optimization (SBO) is often used as a static tool to address a specific one-time question, rather than being dynamically integrated into operational processes to improve day-to-day decisions. However, achieving this integration requires fast delivery of results, which implies short simulation and optimization runtimes. Unfortunately, common SBO algorithms such as genetic algorithms typically require a large number of function evaluations, which translates into many simulation runs. This becomes problematic for computationally intensive simulations, where the time required for the simulation runs and optimization process can be substantial. Bayesian Optimization (BO) methods, which have a higher convergence rate than genetic algorithms (Bull 2011), could offer a more sample-efficient solution for applying SBO in day-to-day decisions.

To make SBO more practical for day-to-day decision-making, this research proposes SimBO (Simulation and Bayesian Optimization), a flexible and component-based framework with well-defined interfaces that applies Bayesian Optimization algorithms to discrete-event simulations. Currently, SimBO is focused on research and aims to investigate how different BO algorithms can be used in simulation problems related to the manufacturing and logistics domain. In the future, the authors envision SimBO evolving into a practical tool that offers default optimization configurations for common problem classes, freeing decision-makers from the need for specific knowledge about SBO algorithms and Bayesian Optimization. Ultimately, SimBO could be offered as an optimization-as-a-service (Bosse et al. 2019) in the cloud, enabling the broad use of novel machine learning-based optimization algorithms.

In the rest of this paper, the authors first give a brief introduction to Bayesian Optimization and explain why it is a promising approach for simulation-based optimization. They then present a selection of BO algorithms that are currently implemented in SimBO. The paper goes on to describe the SimBO architecture and its ability to support a high degree of flexibility and extensibility. The authors then provide implementation details on the current version of SimBO they developed for their research on BO with SBO. The paper concludes with a summary and outlook on future planned activities.

BACKGROUND AND RELATED WORK

Bayesian Optimization (BO), also known as Efficient Global Optimization (Jones et al. 1998), is a method to optimize expensive-to-evaluate and noisy black-box functions (Balandat et al. 2020). The term black-box function in this context means that an analytical expression is not given or hard to define, but we can observe the output of a given input. This holds true for simulations, which in this paper are considered black-box functions in that sense. In general, we aim to solve $max\, f(x)$ with $x \in X$ where $f(x)$ is a black-box function and $X \subset R^d$ a feasible set of d dimensions (Frazier 2018; Balandat et al. 2020). In the context of SBO, $f(x)$ usually represents a performance indicator of the system, like the achieved throughput or accumulated costs. d is defined by the number of influenceable parameters, which are also called *decision variables* or *controls* of the system. Examples are the number of workers at a workstation or the safety stock for a product.

Communications of the ECMS, Volume 37, Issue 1,
Proceedings, ©ECMS Enrico Vicario, Romeo Bandinelli,
Virginia Fani, Michele Mastroianni (Editors) 2023
ISBN: 978-3-937436-80-7/978-3-937436-79-1 (CD) ISSN 2522-2414

Simulations pose special challenges that need to be addressed by the optimization algorithm. These challenges are similar to those found in the problem of hyperparameter tuning of machine learning algorithms, such as deep neural networks. Here, BO algorithms are the state-of-the-art (Frazier 2018). These special characteristics, in addition to the aforementioned noisiness and expensive evaluations, are:

- **Multi-Objectiveness**: SBO problems often face multiple objectives that must be optimized simultaneously without knowing an optimal trade-off in advance.
- **High-Dimensionality**: Dealing with large and complex simulations can lead to hundreds or thousands of parameters or dimensions. A threshold for the number of dimensions to call a problem high-dimensional is not defined. Here, it is assumed that problems with $d >= 20$ can be defined as high-dimensional. This value refers to the soft limit $d < 20$ of "Vanilla BO", mentioned by Frazier (ibid.).
- **Mixed Decision Variables**: Real-world simulations often contain mixed decision variables (continuous and discrete), which can be challenging for algorithms that typically deal with only one variable type (Pelamatti et al. 2018).
- **Constraints**: SBO problems require consideration of input- and output constraints to enable realistic optimization.

There are numerous SBO algorithms that address the aforementioned characteristics and can thus be suitable for the optimization of real-world simulation problems. Some of them are described in Amaran et al. (2016) and Nguyen et al. (2014). However, most popular algorithms, such as CMA-ES (Hansen 2016) and NSGA-II (Deb et al. 2002) are designed for cheap-to-evaluate objective functions and may not be efficient enough for complex simulations with longer runtimes. For such simulations, more efficient algorithms are needed that can achieve good results with a small evaluation-budget (Bull 2011). The authors consider Bayesian Optimization algorithms promising candidates due to their proven high sample-efficiency.

BAYESIAN OPTIMIZATION ALGORITHMS

The ability to optimize expensive-to-evaluate and noisy black-box functions makes BO a popular approach in fields like materials and drug design (Packwood et al. 2017), reinforcement learning (Brochu et al. 2010) and simulations (Schultz et al. 2018). The potential of BO for SBO has long been shown by Chick (2006), among others. Most BO algorithms are computationally expensive (Osborne 2010), but today's powerful and at the same time relatively cheap computation resources allow BO's use in an expanding list of applications. This led to a rapid development and improvement of BO algorithms over the past years (Shahriari et al. 2016; Frazier 2018).

All BO algorithms share the same underlying structure. They have two essential components: the surrogate model (SM) and the acquisition function (ACF). For the surrogate model, mostly Gaussian process regression (GPR) is used. GPR can capture and quantify the uncertainty of the underlying function, which is crucial for stochastic simulations. Further advantages of GPR are explained in Rasmussen et al. (2006).

The surrogate model approximates the black-box-function based on a given set of points (*prior*). In our context of DES, a point is a set of evaluated simulation parameters. The prior model is fed to the acquisition function, which has the task of suggesting the next best set of parameters to evaluate. It works by analyzing the surrogate model and trading off new points that either lie in a promising area of the dimensional space (suspected minimum or maximum), or that are in an area with high uncertainty. Both could hide a potential better value for $f(x)$. The so-found point is evaluated (here: simulated) and the result is handed to the surrogate model to update itself (*posterior*). This process repeats and is called the *optimization loop*. While the loops runs, new points are added and the surrogate model is updated sequentially. The loop stops when the abortion criterion is met. This is typically the case when the evaluation-budget is exhausted or a specific, satisfactory outcome is reached. Like the surrogate model, the acquisition function can also have different implementations depending on the use case. Two widely adopted acquisition functions are *Expected Improvement (EI)* and *Upper Confident Bounds (UCB)* (Frazier 2018; Shahriari et al. 2016).

Over the last few years, several algorithms from the family of Bayesian Optimization were published. Each addresses the challenges of SBO with different approaches. Based on a first feasibility assessment, the authors selected a list of currently six algorithms to be implemented in SimBO. In the following, the six algorithms are briefly outlined, with emphasis on their basic approach. Further details can be obtained from the references given in each subsection.

Table I: Capabilities of selected BO algorithms regarding the challenges of simulation-based optimization.

Algorithm	Multi-Objectiveness	High-Dimensionality	Mixed Decision Variables	Constraints
GPEI	no	no	partial	yes
TurBO	no	yes	partial	yes
MORBO	yes	yes	partial	yes
SAASBO	no	yes	partial	no
qNEHVI	yes	partial	partial	yes
qNEHVI-SAASBO	yes	yes	partial	no

GPEI is short for Gaussian Process - Expected Improvement. It is similar to the EGO algorithm, presented by Jones et al. (1998). It is used for continuous parameters with a soft limit of 20 Dimensions (Frazier 2018) and can handle single objective problems. It is also called "Vanilla BO" (Feurer et al. 2018) because it implements the idea of BO in its basic form.

TuRBO is short for Trust Region Bayesian Optimization (Eriksson et al. 2019). Instead of one large model, it fits smaller local GP models (trust regions) in promising regions of the search space and allocates samples across these models globally. Depending on their success, trust regions can shrink, grow, or ultimately die and restart in a different area of the dimensional space. TuRBO is used for high-dimensional, continuous, single-objective problems. There is an enhancement for multi-objective problems called **MORBO** (Daulton et al. 2022a).

SAASBO stands for Sparse Axis Aligned Subspace Bayesian Optimization and focuses on high dimensional, continuous, single-objective problems. It assumes a hierarchy of feature relevance and tries to identify these features (parameters) during the optimization, resulting in a more useful model for exploration and exploitation of the most important parameters (Eriksson et al. 2021b).
qNEHVI-MOBO is an algorithm for multi-objective Bayesian Optimization (**MOBO**). Instead of a single best point, the result in multi-objective optimization is a Pareto front for the objectives, where for all points on this front it holds true that none of the objectives can be improved without compromising another one. MOBO uses Gaussian processes (GP) to model the objectives, and draws samples based on the expected hypervolume improvement (**EHVI**) criterion. It allows parallel sampling (**q**) and considers noisy objectives (**N**) (Daulton et al. 2021).

Furthermore, combinations of different surrogate models and acquisition functions were proposed, such as **qNEHVI-SAASBO** (Eriksson et al. 2021a). For dealing with parameter spaces with mixed decision variables (continuous and discrete/categorical), typically the one-hot-encoding approach is used, but other approaches are discussed. The interested reader is referred to Garrido-Merchán et al. (2020), Yang et al. (2019), and Daulton et al. (2022b).

While the author's of most of the optimization algorithms mentioned provide example implementations, they greatly differ in the way they can be used and need customization to be applied to simulations, for which the same can be said. SimBO aims to consolidate the optimization algorithms as well as the simulation models and create a common interface for both.

SIMBO

Components Overview

SimBO consists of a user interface, a database, and the SimBO core. Moreover, SimBO requires a simulation model (or any black-box function that can be evaluated) to optimize (figure 1).

User Interface

SimBO has a web-based user interface that allows users to manage optimization experiments. An optimization experiment involves optimizing a black-box function (in this case, a simulation model) using a specified optimization algorithm, for which a set of hyperparameters must be configured. The experiment also includes parameters such as an abortion criterion.
Once set up, the user can schedule, run, pause, and stop an experiment as well as view the optimization status while the experiment is running and the optimization results after the experiment has finished. All information about an experiment, including the status and results, is stored in the SimBO database.

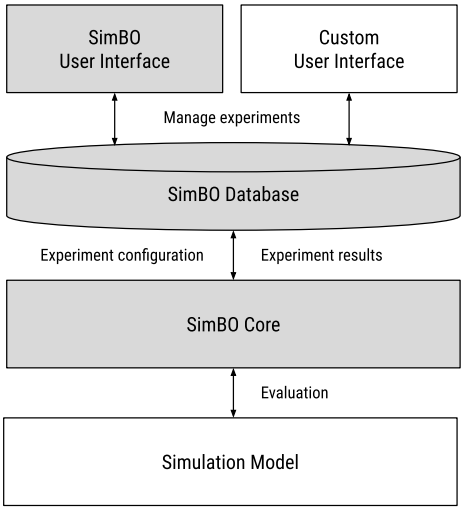

Figure 1: Components of the SimBO architecture and their interaction.

Database

SimBO utilizes a NoSQL database to store experiment configurations and results. This approach offers greater flexibility than a rigidly structured relational database and can easily handle data in JSON format. This flexibility is particularly important because the optimization algorithms and simulation models managed in the database exhibit significant heterogeneity in their attributes. Additionally, the database must accommodate the inclusion of new algorithms and simulation models, whose attributes are unknown in advance. The JSON format is used mainly for storing the experiments and their trial history.
The SimBO database acts as an intermediary between the SimBO core and the user interface (UI). It is worth noting that, although SimBO comes with a UI, it is possible to implement and use a different UI as long as it is compatible with the SimBO database. In the future, a custom UI may prove useful for practitioners who seek to incorporate advanced optimization techniques into their day-to-day decision-making processes. These users may not require the full range of functionalities provided by the SimBO UI, but instead prefer a pre-configured set of use-cases that can be executed efficiently.

Core

The SimBO core comprises the SimBO manager, experiment runner, and two types of abstract bridges, one to connect to different optimization algorithms and another to interface with the simulation models (see figure 2). The SimBO manager is the entry point for running an experiment. Its primary task is to retrieve information about the experiment from the SimBO database and instantiate the appropriate experiment runner. The experiment runner receives the information about the experiment and uses it to create the required bridges for the specific algorithm and simulation model.
There are two types of experiment runners: optimization-driven and simulation-driven. The optimization-driven experiment runs in a closed loop, which is suitable if the simulation can be synchronously executed from the SimBO pro-

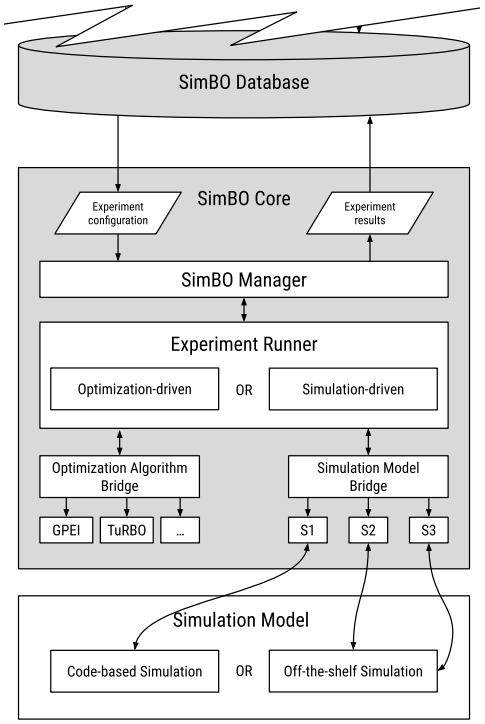

Figure 2: A detailed view of the SimBO core architecture.

gram code. An example is a simulation model written in the Python-based SimPy library (Matloff 2008). In contrast, the simulation-driven experiment runs asynchronously using an ask-and-tell interface. This approach is suitable if the simulation model is executed by an off-the-shelf simulation tool with a custom graphical user interface for running experiments. Here, SimBO cannot directly initiate a simulation run, but it can provide a web-service-based interface for the simulation tool to request new parameterization sets for evaluation. A critical prerequisite is that the simulation tool offers a mechanism to extend the experimentation capabilities, such as through the implementation of a custom add-in. An example is Simio with its C#-API that allows writing experimentation add-ins that can be loaded in the UI and take over the default experimentation process in Simio while still utilizing Simio's UI.

In the optimization-driven case, the optimization component controls the optimization process and calls the simulation model during the run. In contrast, in the simulation-driven scenario, the simulation component is in control and actively asks the optimization component for points to evaluate, and afterward reports (*tells*) the results. The chosen approach depends on the type of simulation model and the software it was created with.

Regardless of the type of experiment runner, optimization-driven or simulation-driven, two so-called *bridges* are required to conduct an experiment: one for the optimization algorithm, and one for the simulation model. Both bridges provide access to the respective components through an abstract class with a standardized interface. Through this encapsulation, new algorithms and simulation models can be added easily.

An **optimization algorithm bride** requires the following three functions:

- **Initialize**: Define the search space and objectives and initialize the optimization model by generating random parametrization sets, using Sobol-sequences (Sobol 1967), which are evaluated by the simulation.
- **Suggest**: Suggest one or more parametrization sets, based on the current surrogate model and acquisition function. The parametrization sets are part of the same *trial*. A single parametrization from this set is called a *candidate*.
- **Complete**: Receive the evaluation results for a trial (or a subset of candidates) and hand it to the model. It should be noted that at this step the model is not refitted (yet).

The **simulation model bridge** exposes an evaluation endpoint to synchronously evaluate a parameterization set for headless and code-based simulation models. That is, the experiment runner calls the endpoint, which is typically a function, and waits for the simulation to return the results. For off-the-shelf simulation tools, the simulation model bridge does not provide a such an endpoint for evaluation because here, the simulation is the driver and does not get called. Instead, the simulation model calls the exposed web-services of the optimization bridge in an ask-and-tell manner (suggest-compelete cycles). Additionally and for both cases, the simulation model bridge provides access to the important information about the simulation model, such as its parameters (e.g., name and type), the objectives and whether they should be minimized or maximized, and optionally constraints that must be considered.

Simulation Model

As described above, SimBO can handle two types of simulation models: code-based simulations and simulations created with an off-the-shelf simulation tool. Code-based simulations are directly written in a programming language and can therefore be executed by the optimization engine in a synchronous manner. They typically expose a function with corresponding parameters for the simulation model's controls. Off-the-shelf simulations tools cannot be run in that manner and need an additional interface to communicate with the optimization engine. Here, the logic is reversed, and the simulation tool is responsible for the optimization process.

Simulation tools like Simio allow for running so-called *headless* simulations, which could be considered a third type. However, a headless simulation behaves similar to a code-based simulation. Instead of a function call, a headless simulation is typically an executable file that can be run via the command line. In this paper, both types are subsumed under the code-based simulation models.

Experiments

To run an optimization, the user must first define an experiment, preferably using the SimBO UI. An experiment requires the specification of a simulation model and an optimization algorithm with all its hyperparameters. For the experiment, the user is required to set an abortion criterion with three options to choose from: evaluation budget, threshold or desired improvement. Setting an evaluation

budget is the most common. Here, the optimization will run for the specified number of simulation runs. Setting a threshold means the user can define a result which should be achieved at minimum. When reached, the optimization will terminate. It could well be that the threshold is never reached. To have a fallback, the user must additionally define an evaluation budget. As the third option, the user can express a desired relative improvement I in percent with a number of trials n. In this case, the optimization will terminate when the desired improvement I was not reached over the last n trials.

Once created, the user can run the same experiment setting multiple times. This is particularly useful for recurring optimizations of the same simulation model and the same optimization configuration.

The user can monitor the execution of an experiment in real-time using the SimBO UI. Here, the user can see the progress in terms of completed simulation runs or the remaining evaluation budget. Furthermore, the UI displays an overview of the most important performance metrics for each experiment, including the current best solution. Currently, the user has to check manually if an experiment is finished. Because experiments can run for many days, it is planned to implement a notification service, for example via e-mail, to inform the user about important events.

After an experiment is finished, the user can open the UI and view a compact version of the experiment result, such as the best point found or the total run time. The experiment data along with a standard PDF-report including data visualizations is provided as a download.

PROTOTYPICAL IMPLEMENTATION

The SimBO architecture described in the previous section has been implemented for evaluation and research purposes. In this section, the implementation details for the user interface, the database, the SimBO core, and the simulation models are described.

User Interface

The prototype uses a web-frontend based on Vue, a modern JavaScript framework (You 2023). It provides a minimal set of masks to create and manage experiments. The website is self-hosted and currently inaccessible from outside the university's network.

Database

The SimBO database is implemented using Firestore, a NoSQL database included in the Google Firebase cloud platform (Google Inc. 2023). It enables fast implementation cycles with an easy-to-use API and does not require any manual installation or configuration. For applications with data privacy requirements, one could use a self-hosted, open-source alternative such MongoDB (MongoDB Inc. 2023).

Core

The SimBO core is running on a high-performance cluster owned by the University of Applied Sciences in Osnabrück, Germany. The cluster consists of multiple nodes, each with a 40 GB GPU (Nvidia A100), 128-core CPU and 512 GB RAM. Each optimization run is executed on a single node.

The SimBO core is fully Python-based. For the implementation of the optimization algorithms, SimBO builds on top of BoTorch (Balandat et al. 2020), a Python-library for Bayesian optimization based on the popular machine learning library PyTorch (Paszke et al. 2019). With BoTorch at its core, SimBO can leverage GPU computation when needed. In addition to the BO algorithms mentioned in section "Bayesian Optimization Algorithms", the SimBO prototype implements two genetic algorithms, CMA-ES (Hansen et al. 2023) and NSGA-II (Blank et al. 2020), for benchmarking purposes. Further, the Sobol-algorithm, a quasi-random point generator, is included as a baseline for the convergence-rate benchmarking.

Simulation Models

For code-base simulations, we use SimPy, a python library for discrete-event-simulation (Matloff 2008). The simulations are either executed on the same hardware as the optimization (for code-based simulations) or on a virtual machine (Windows 10, octa-core 3.1 GHz, 16 GB RAM) in the case of the Simio simulation software.

For the Simio software, an experiment add-in was developed to create an interface to communicate with SimBO. The add-in manages the optimization run by calling the web-service endpoints mentioned earlier. All endpoints are provided by the simulation-driven experiment runner. All Simio simulation models are executed on the aforementioned Windows-based virtual machine.

Current use cases under examination are the optimization of material requirements planning simulation and the optimization of mold procurement to minimize the waste in the injection molding industry.

Application Scenarios

Bayesian optimization is a versatile technique that can be applied to a variety of scenarios involving black-box problems, where inputs and outputs are known and a mathematical relationship between them can be assumed. The authors focus on discrete event-driven simulations in manufacturing and logistics. Specifically, they use SimBO to address two problems: Material Requirement Planning (MRP) simulations and resource allocation for waste reduction in the plastics industry. While the latter problem is still at an early stage and won't be discussed further, the optimization of the MRP simulation is well advanced, and the authors plan to publish results in the coming months.

The MRP optimization problem is characterized by its high dimensionality, which involves hundreds of parameters, as well as its multi-objective nature, which considers both costs and service level. Additionally, there's a hierarchical linkage between the underlying materials and the corresponding parameters. In the optimization process, safety stock and safety lead time are optimized per product to minimize cost while providing a specific service level, or alternatively, maximize service level in a multi-objective scenario.

The increasing popularity of Bayesian optimization has led more researchers and practitioners to apply this technique to their simulations (Shahriari et al. 2016). For instance, Kiuchi et al. (2020) use Bayesian optimization to optimize inventory levels in agent-based supply-chain simulations, while Chepiga et al. (2023) apply it to optimize process pa-

rameters in stainless steel production.

CONCLUSION

This paper introduced SimBO, a framework designed for simulation-based optimization using Bayesian Optimization. Its purpose is to streamline optimization experimentation by consolidating the optimization and simulation components and offering a user-friendly interface to pair different algorithms with different simulation models. The algorithms utilized in SimBO are tailored to address various aspects of simulations such as mixed decision-variables, high dimensionality, and multi-objectiveness. The algorithms are built on top of the BoTorch framework wherever possible to take advantage of GPU acceleration. SimBO can handle both code-based (including headless) and off-the-shelf simulations.

Currently, SimBO is primarily focused on facilitating research into the suitability of modern optimization algorithms for SBO. As an example, the authors are utilizing SimBO in an ongoing research project that involves applying various BO algorithms to a simulation for material requirements planning (MRP). In this project, SimBO serves as the orchestrator for the optimization and simulation components, automating all experiments conducted.

In the future, the aim is to transform SimBO into a tool that is production-ready, and possibly cloud-based, with the intention of serving both researchers and practitioners. The next steps in the development of SimBO will be to provide users with the ability to create custom user interfaces to interact with the system. Additionally, SimBO will integrate more optimization algorithms typically applied in SBO, as well as create additional interfaces for widely adopted simulation software available on the market.

The code for the current stage of SimBO can be found at: `https://github.com/pehzet/SimBO`.

REFERENCES

Amaran, Satyajith, Nikolaos V Sahinidis, Bikram Sharda, and Scott J Bury (2016). "Simulation optimization: a review of algorithms and applications". In: *Annals of Operations Research* 240, pp. 351–380.

Balandat, Maximilian, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy (2020). "BoTorch: A framework for efficient Monte-Carlo Bayesian optimization". In: *Advances in neural information processing systems* 33, pp. 21524–21538.

Blank, Julian and Kalyanmoy Deb (2020). "Pymoo: Multi-objective optimization in python". In: *IEEE Access* 8, pp. 89497–89509.

Bosse, Sascha, Abdulrahman Nahhas, Matthias Pohl, and Klaus Turowski (2019). "Towards an Automated Optimization-as-a-Service Concept." In: *IoTBDS*, pp. 339–343.

Brochu, Eric, Vlad M Cora, and Nando De Freitas (2010). "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning". In: *arXiv preprint arXiv:1012.2599*.

Bull, Adam D (2011). "Convergence rates of efficient global optimization algorithms." In: *Journal of Machine Learning Research* 12.10.

Chepiga, Timur, Petr Zhilyaev, Alexander Ryabov, Alexey P Simonov, Oleg N Dubinin, Denis G Firsov, Yulia O Kuzminova, and Stanislav A Evlashin (2023). "Process Parameter Selection for Production of Stainless Steel 316L Using Efficient Multi-Objective Bayesian Optimization Algorithm". In: *Materials* 16.3, p. 1050.

Chick, Stephen E. (Dec. 2006). "Bayesian Ideas and Discrete Event Simulation: Why, What and How". In: *Proceedings of the 2006 Winter Simulation Conference*. ISSN: 1558-4305, pp. 96–106. DOI: 10.1109/WSC.2006.323042.

Daulton, Samuel, Maximilian Balandat, and Eytan Bakshy (2021). "Parallel bayesian optimization of multiple noisy objectives with expected hypervolume improvement". In: *Advances in Neural Information Processing Systems* 34, pp. 2187–2200.

Daulton, Samuel, David Eriksson, Maximilian Balandat, and Eytan Bakshy (2022a). "Multi-objective bayesian optimization over high-dimensional search spaces". In: *Uncertainty in Artificial Intelligence*. PMLR, pp. 507–517.

Daulton, Samuel, Xingchen Wan, David Eriksson, Maximilian Balandat, Michael A Osborne, and Eytan Bakshy (2022b). "Bayesian optimization over discrete and mixed spaces via probabilistic reparameterization". In: *arXiv preprint arXiv:2210.10199*.

Deb, Kalyanmoy, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan (2002). "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE transactions on evolutionary computation* 6.2, pp. 182–197.

Eriksson, David, Pierce I-Jen Chuang, Samuel Daulton, Peng Xia, Akshat Shrivastava, Arun Babu, Shicong Zhao, Ahmed Aly, Ganesh Venkatesh, and Maximilian Balandat (2021a). "Latency-aware neural architecture search with multi-objective bayesian optimization". In: *arXiv preprint arXiv:2106.11890*.

Eriksson, David and Martin Jankowiak (2021b). "High-dimensional Bayesian optimization with sparse axis-aligned subspaces". In: *Uncertainty in Artificial Intelligence*. PMLR, pp. 493–503.

Eriksson, David, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek (2019). "Scalable global optimization via local bayesian optimization". In: *Advances in neural information processing systems* 32.

Feurer, Matthias, Benjamin Letham, Frank Hutter, and Eytan Bakshy (2018). "Practical transfer learning for bayesian optimization". In: *arXiv preprint arXiv:1802.02219*.

Fowler, John, Sondoss El Sawah, and Hasan Hüseyin Turan (2022). "Recent advances in simulation-based optimization for operations research problems". In: *Annals of Operations Research*, pp. 1–2.

Frazier, Peter I (2018). "A tutorial on Bayesian optimization". In: *arXiv preprint arXiv:1807.02811*.

Garrido-Merchán, Eduardo C and Daniel Hernández-Lobato (2020). "Dealing with categorical and integer-valued variables in bayesian optimization with gaussian processes". In: *Neurocomputing* 380, pp. 20–35.

Google Inc. (2023). *Firebase Cloud Service*. Accessed: 2023-02-21.

Hansen, Nikolaus (2016). "The CMA evolution strategy: A tutorial". In: *arXiv preprint arXiv:1604.00772*.

Hansen, Nikolaus, yoshihikoueno, ARF1, Gabriela Kadlecová, Kento Nozawa, Luca Rolshoven, Matthew Chan, Youhei Akimoto, brieglhostis, and Dimo Brockhoff (Jan. 2023). *CMA-ES/pycma: r3.3.0*. Version r3.3.0. DOI: 10.5281/zenodo.7573532. URL: https://doi.org/10.5281/zenodo.7573532.

Jones, Donald R, Matthias Schonlau, and William J Welch (1998). "Efficient global optimization of expensive black-box functions". In: *Journal of Global optimization* 13.4, p. 455.

Kiuchi, Atsuki, Haiyan Wang, Qiyao Wang, Takahiro Ogura, Tazu Nomoto, Chetan Gupta, Takaharu Matsui, Susumu Serita, and Chi Zhang (2020). "Bayesian Optimization Algorithm with Agent-based Supply Chain Simulator for Multi-echelon Inventory Management". In: *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, pp. 418–425. DOI: 10.1109/CASE48305.2020.9216792.

Matloff, Norm (2008). "Introduction to discrete-event simulation and the simpy language". In: *Davis, CA. Dept of Computer Science. University of California at Davis. Retrieved on August* 2.2009, pp. 1–33.

MongoDB Inc. (2023). *MongoDB*. Accessed: 2023-02-21.

Nguyen, Anh-Tuan, Sigrid Reiter, and Philippe Rigo (2014). "A review on simulation-based optimization methods applied to building performance analysis". In: *Applied energy* 113, pp. 1043–1058.

Osborne, Michael A (2010). "Bayesian Gaussian processes for sequential prediction, optimisation and quadrature". PhD thesis. Oxford University, UK.

Packwood, Daniel et al. (2017). *Bayesian optimization for materials science*. Springer.

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. (2019). "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32.

Pelamatti, Julien, Loıc Brevault, Mathieu Balesdent, El-Ghazali Talbi, and Yannick Guerin (2018). "How to deal with mixed-variable optimization problems: An overview of algorithms and formulations". In: *Advances in Structural and Multidisciplinary Optimization: Proceedings of the 12th World Congress of Structural and Multidisciplinary Optimization (WCSMO12) 12*. Springer, pp. 64–82.

Rasmussen, Carl Edward and Christopher K. I. Williams (2006). *Gaussian processes for machine learning*. en. Adaptive computation and machine learning. OCLC: ocm61285753. Cambridge, Mass: MIT Press. ISBN: 978-0-262-18253-9.

Riley, Linda Ann (2013). "Discrete-event simulation optimization: a review of past approaches and propositions for future direction." In: *SummerSim*, p. 47.

Schultz, Laura and Vadim Sokolov (2018). "Bayesian optimization for transportation simulators". In: *Procedia computer science* 130, pp. 973–978.

Shahriari, B., K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas (2016). "Taking the Human Out of the Loop: A Review of Bayesian Optimization". In: 104. ISSN: 1558-2256. DOI: 10.1109/JPROC.2015.2494218.

Sobol, Il'ya Meerovich (1967). "On the distribution of points in a cube and the approximate evaluation of integrals". In: *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki* 7.4, pp. 784–802.

Yang, Kaifeng, Koen van der Blom, Thomas Bäck, and Michael Emmerich (2019). "Towards single-and multi-objective Bayesian global optimization for mixed integer problems". In: *AIP Conference Proceedings*. Vol. 2070. 1. AIP Publishing LLC, p. 020044.

You, Evan (2023). *VueJS - The Progressive JavaScript Framework*. Accessed: 2023-02-21.

AUTHOR BIOGRAPHIES

Philipp Zmijewski Philipp Zmijewski is a PhD student and research assistant at the University of Applied Sciences Osnabrück. In his doctoral thesis, he is exploring the application of Bayesian Optimization to discrete-event simulation.

Nicolas Meseth Nicolas Meseth is a Professor of Information Systems at the University of Applied Sciences Osnabrück. His research focuses on applications of AI and machine learning in the food industry.