Automated Machine Learning for Business Decision Simulation

Reinhard Bauer[1] and Tyler Marangi[2]

[1] Karlsruhe University of Applied Sciences
reinhard.bauer@h-ka.de
[2] SAP SE
tyler.marangi@sap.com

Abstract. In many industries it is common to support business decisions with the help of simulation models. Such models can be used to predict the outcome of a series of decisions. Often, the number of possible options is high, making it difficult to identify good candidates for simulation. Optimization models can help to find such candidates. Building optimization models can be an arduous task for a variety of reasons, such as computational complexity, algorithmic complexity, or required expertise. Many times, simulation models are readily available.
We propose to use black-box optimization techniques on simulation models to identify good business decisions. This allows generic optimization strategies to be applied to simulation models without additional effort or knowledge. There are numerous existing research papers on black box optimization. A traditional application of such routines is parameter optimization for algorithm tuning, e.g. solvers for mixed-integer linear programming. In recent years, a major focus has shifted to automated machine learning. The challenge here is to find good hyper-parameters for machine learning approaches or even to choose the best strategy automatically.
We sketch different approaches to automated machine learning and how to apply them on simulation models. To demonstrate the benefit of our approach, we create a use-case for energy optimization in a commercial facility, such as an office building. Finally, the performance of the different approaches is evaluated in an experimental study.

1 Introduction

In recent years, improved data availability resulted in demand for better use of the data collected. Business decision support systems have a long history, often originating in the field of operations research. Both simulation as well as optimization models are well-established tools to support business decisions. While most applications allow for custom-tailored, hand-made optimization models, this approach often is too costly to apply in commercial practice.

In case a simulation model already exists for a concrete application, we propose to use techniques from the field of black-box optimization / automated machine learning in order to automatically add optimization capabilities to this model. The general workflow of our approach can be seen in Figure 1.

The business decisions are included as parameters of the simulation model (visualized as red boxes) and have to be fixed before starting to simulate. Once those parameters are determined, the simulation phase starts. This involves one simulation run for a deterministic model and a multitude of runs for a stochastic model. After simulating, the simulation outcome is evaluated. To that end, an objective function is specified which models targets like profit, cost or throughput.

The black-box optimization routine can interact with the simulation model only by setting the parameters (specifying the input of the function to be optimized) and reading the resulting objective value (evaluating the output of the function). In a loop, the black-box optimization routine iteratively identifies new promising candidate parameter sets, triggers the simulation run and reads back the objective value. New promising candidates are then identified and the loop starts all over. The actual way in which the next candidates are identified differs between the approaches to black-box optimization. To the best of our knowledge, this approach has not been applied before to business decision simulation.

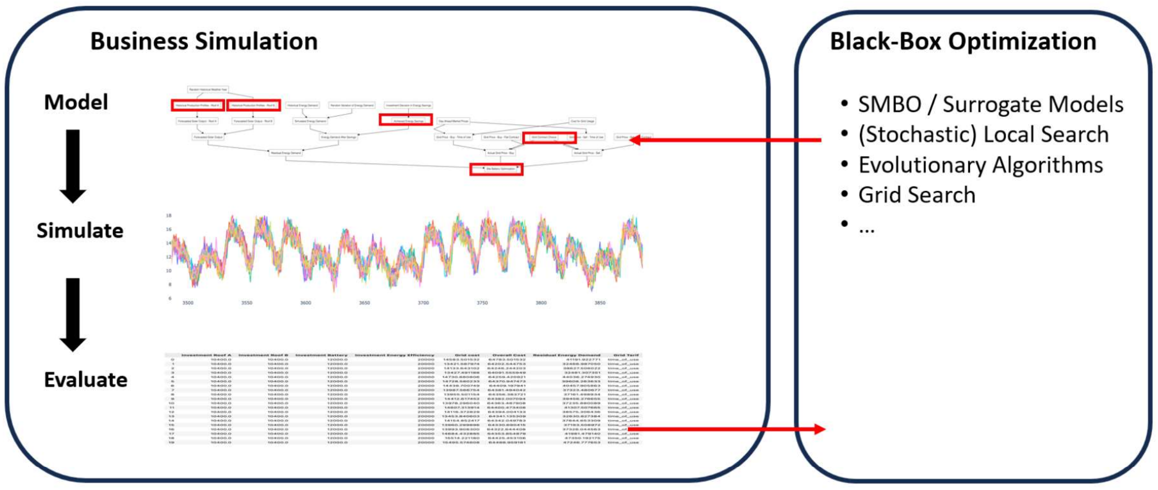


Fig. 1. Interaction between simulation and black-box optimization routine.

Related Work. *Black-box optimization* is a sub-field of mathematical optimization where a function $f : D \to \mathbb{R}$ is to be numerically optimized without further strong assumptions on properties of f, in particular without using derivatives. In a typical setup, the only way to obtain information about f is to evaluate f for a given input value, which in that context often is painstakingly time-consuming. There exist numerous methods for black box optimization, an overview can be found in [1]. An important distinction is between *model-free* and *model-based* methods. The first class encompasses most of the common (meta-)heuristics such as *random search*, *evolutionary algorithms*, *simulated annealing*, *hill-climbing*, *direct search* and many more. The common property of the second class is to build an approximation model of f, called the *surrogate*. The underlying assumption here is, that f is somehow smooth (which arguably is an assumption for most of black-box optimization). Good results are reported for surrogates based on gaussian processes and random forests.

Automated machine learning (AutoML) (see [2] for an overview) is the task to automatically identify good hyperparameters for machine learning methods. Many methods in AutoML do not exploit special knowledge on the technique to be trained and hence can be seen as a special case of black-box optimization. An important algorithm stemming from this community is *SMAC* [3]. Many AutoML-methods are readily available as open-source libraries such as *Ray* (`ray.io`), *Optuna* (`optuna.org`), *SMAC3* (`github.com/automl/SMAC3`), *Scikit-Learn* (`scikit-learn.org`).

It already is widespread practice to combine operations research with artificial intelligence. For a recent survey paper, see [4].

2 Black-Box Optimization for Simulation Models

We consider the following algorithm classes: Stochastic search, local search, grid search, a genetic algorithm, a simple variant of sequential model-based optimization using random forests as surrogates and some highly tuned model-based algorithms offered as part of the SMAC3-library (for details see github.com/automl/SMAC3). For each approach (except grid search) we allow a predefined budget of n simulation runs.

2.1 Grid search

For each parameter p_i, a set of possible sample points is selected. Grid search is an exhaustive search over all possible combinations of those sample points.

For discrete parameters, we select all possible values as sample points. For continuous parameters and a given number of samples per parameter, we select sample points equally spaced over the entire domain (starting and ending at the domain boundaries).

In many real-world settings, grid search is too slow to apply. However, we use it to approximate the optimal solution of our use case.

2.2 Stochastic (or random) search

Stochastic search evaluates n random instances for which each parameter has been chosen uniformly at random in the respective domain. This approach can be used as a baseline for comparison with other more sophisticated strategies.

2.3 Stochastic local search

Stochastic local search starts with a random instance (selected as in stochastic search) as an *active instance*. Then, new instances in a *neighborhood* around the active instance are sampled randomly and are subsequently evaluated. If an instance S is found to be better than the currently active instance, it becomes the new active instance. The approach stops after n instances have been evaluated.

We apply *restarts*: If, for some iterations, no improvement has been realized around the currently active instance, a new active instance is chosen uniformly at random.

We allow the target value to deteriorate: With a certain probability, a worse parameter set can also become the new active instance.

When sampling from the neighborhood, we proceed as follows: For a continuous parameter p_i with domain $[l_i, u_i]$ we sample from a normal distribution whose mean is the corresponding parameter value of the active solution and whose variance is $0.1(u_i - l_i)$. We clip values to the domain bounds if necessary. When sampling a discrete parameter, we keep the current value with probability $1 - c$ and choose a new value uniformly at random with probability c where c is a tuning parameter.

2.4 Evolutionary algorithms

The basics of evoluationary algorithms can be found in any introductionary textbook on the matter. We use the following specifics:

– Tournament selection with one elitist individuum in the next generation.
– Crossover: For each parameter p_i, the value is chosen uniformly at random from one of the two parents.
– Mutation uses the neighborhood definition of the local search approach: The routine to mutate an individuum is the same as to randomly identify a new instance in the neighborhood of the currently active solution.

2.5 Model-based approach using random-forests as surrogates

We use a minimalistic model-based approach:

– Exploration is done only in an introductionary phase. Here, for a number of samples k, a Sobol sequence [5] is constructed with dimensionality given by the number of parameters. Afterwards, the actual values of the discrete parameters are obtained by rounding. Each instance is evaluated by a simulation run.
– The following exploitation phase comprises $n - k$ iterations: At each step, first a random forest is trained on all instances evaluated so far. Features are the respective parameters. Targets are the corresponding objective values.
  Then, a Sobol sequence with 10.000 samples is created as described above. The random forest is used to estimate the objective value of each sample. The most promising candidate then is evaluated by a simulation run.

2.6 SMAC3

We test SMAC3 as a representative of a state-of-the-art framework for black-box optimization.

3 Application Scenario

3.1 Overview

As a use case to test the efficiency of our approach, we model the decisions involved in managing the energy aspects of a mid-sized office building. These comprise long- and short-term decisions. The long-term decisions are:

– The building has two roofs on which photovoltaic panels can be installed. Orientation and inclination of both roofs differ as does the maximum number of panels that can be installed. The decision to be taken is, for each roof, the photovoltaic capacity to build.
– It is possible to invest in energy efficiency measures, namely more efficient lighting and more efficient ventilation. Exactly one out of four options is to be chosen: Do not invest, invest only in lighting, invest only in ventilation, invest in lighting and ventilation.
– A battery energy storage system can be purchased. The power rating of the battery is fixed. The exact capacity of the system is to be decided.

– Two contract types are available for electricity procurement: A flat tariff as well as a tariff depending on the price of the day-ahead market.

The time-granularity of the model is one hour. From an operational point of view, the following decision must be taken for each hour of the model horizon:

– Should the battery charge, discharge or do nothing (and to which extent)?

Furthermore, the energy demand of the building is modeled. This constitutes the simulation part of the model:

– The actual power demand of the building is modeled based on historic energy consumption with volatility added.

Annualized cost is given for all investments. The objective is to minimize the expected energy cost over a given horizon including the annualized investment and the income of selling power to the grid.

3.2 Detailed Description

The model horizon consists of hourly periods $1, \ldots, T$. The model incorporates stochastic elements and hence is simulated n times for a given number n. The model outcome is the mean energy cost over all simulation runs. The overall model can be broken down into three parts: Residual load calculation, contract type/tariff choice and the battery model. All long-term decisions fixed, the former two parts can be calculated independently. Their outcome is an input to the battery model which decides the operational decisions.

Residual load. An estimate for the future hourly electrical energy demand is given as input data. We assume that investing in an energy efficiency measure reduces hourly demand by a fixed relative value. A jump diffusion process is added to the expected energy demand to account for the volatility of actual electricity consumption. Finally, expected solar production is subtracted. To that end, a historic year in the range 2005 to 2016 is chosen uniformly at random. The *expected solar power production* equals the power production that the considered number of pv-panels with same location and orientation would have produced in the same historic range. *Residual load* is the actual load after adjusting for the investments in energy efficiency and expected solar power production. Please note that residual load can be negative, resulting in *excess energy.* Residual load is handed over as input to the battery model.

Grid Tariff. There are two different contract types: Firstly, a *flat tariff* where the price k_t for buying energy from the grid is equal for all hours t, as is the price v_t for selling energy. Secondly, a *flexible tariff* where those prices are individual for each hour. We assume those prices are known for the entire model horizon.

Battery model. The battery is modelled using a mixed-integer linear program. Perfect foresight over the model horizon is assumed. Battery ageing is not taken into account (neither cyclic ageing, nor calendar degradation). Influence of temperature or other environmental factors are not taken into account. Input values are

– The prices k_t and v_t for buying/selling energy from/to the grid at hour t.
– The residual load d_t at hour t. Positive values mean demand, negative values mean excess energy.

– The battery efficiency η (we assume the same efficiency for charging and discharging).
– The battery capacity c_{max}.
– The battery power s_{max}.

Decision variables are

– Amount of energy p_t^+ and p_t^- bought and sold from and to the grid at hour t, respectively.
– Amount of energy s_t^+ charged into the battery (before considering efficiency) at hour t.
– Amount of energy s_t^- discharged from the battery (after considering efficiency) at hour t.

The resulting linear program is

$$\text{minimize} \sum_{t=1}^{T} k_t p_t^+ - v_t p_t^-$$

subject to

$$\begin{aligned}
p_t^+ + s_t^- &= d_t + p_t^- + s_t^+ && \text{for } t = 1, 2, \ldots, T && (1)\\
s_{t+1} &= s_t + s_t^+ \eta - s_t^- / \eta && \text{for } t = 1, 2, \ldots, T-1 && (2)\\
s_t &\leq c_{max} && \text{for } t = 1, 2 \ldots, T && (3)\\
s_t^+ &\leq s_{max} \cdot charging_t && \text{for } t = 1, 2 \ldots, T && (4)\\
s_t^- &\leq s_{max} \cdot discharging_t && \text{for } t = 1, 2 \ldots, T && (5)\\
charging_t + discharging_t &\leq 1 && \text{for } t = 1, 2 \ldots, T && (6)\\
0 &\leq p_t^-, p_t^+,\ s_t,\ s_t^+,\ s_t^- && \text{for } t = 1, 2 \ldots, T && (7)\\
charging_t,\ discharging_t &\in \{0, 1\} && \text{for } t = 1, 2 \ldots, T &&
\end{aligned}$$

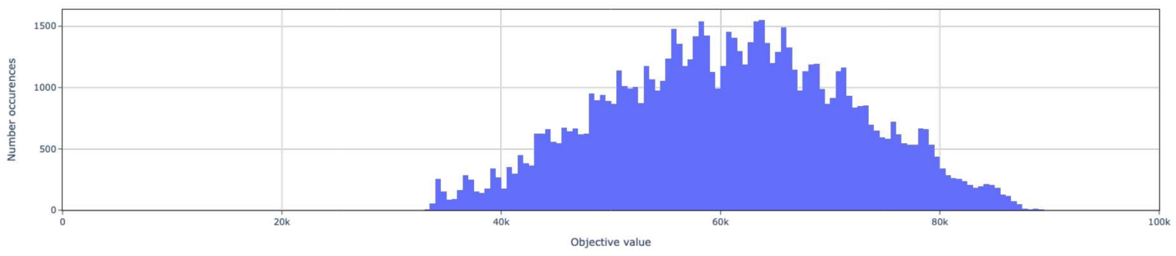
The modeled aspects of each equation are as follows: (1) Equality of supply and demand, (2) state of charge, (3) maximum capacity, (4) indicate charging, (5) indicate discharging, (Either charge or discharge), (7) non-negativity. Note, that the constraints to forbid simultaneous charging/discharging are only necessary when energy prices may be negative.

Potential Enhancements. It is worthwhile to add volatility to the flexible power tariff, incorporate battery ageing and model the energy demand more detailedly.

4 Experimental Evaluation

We conduct a computational study to assess the quality of the black-box optimization routines. The problem to be solved is the scenario explained in the last section. All experiments were performed on a MacBook Air with M1 Processor and 8GB RAM, running Python 3.8.12. Mixed-integer linear programs were solved with the COIN-OR Branch-and-Cut Solver (`www.coin-or.org`).

Grid Search. As the global optimum is unknown, we perform a grid search as a way to estimate the best possible objective value as well as the variability of the problem. Discrete decisions (efficiency investment and contract choice) are considered in their entirety. For each continuous parameter (installed photovoltaic capacity on each roof, battery sizing), ten equidistant values were chosen. The required run-time was 16h 26min, the best solution found had a target value of 33303. The distribution of all evaluated parameter choices is as follows:



Performance of the black-box routines. The run-time of all approaches considered was dominated by the time required for simulation. The overhead for the optimization algorithm itself was negligible, even for the surrogate-based approaches. In this study we allow a time-budget of 50 simulation runs per approach. Some methods (e.g. the evolutionary strategy) allow for parallel runs, while others (e.g the surrogate-based algorithms) must be executed sequentially. Counted sequentially, the overall run-time of each strategy roughly is 1:40 min.

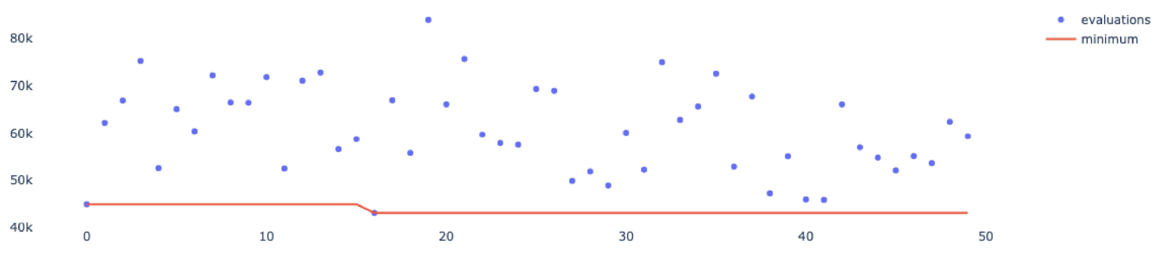
Solution quality of all black-box routines is near-optimal (assuming that the grid-search is dense enough to make that judgement). The individual best objective values found are

Grid Search	33303
Stochastic Local Search	33979
Evolutionary Algorithm	34184
SMBO - Random Forest	34217
SMAC3 - Hyperparameter Optimization Facade	34419
SMAC3 - Black Box Facade	34608
Stochastic Search (for comparison)	43037

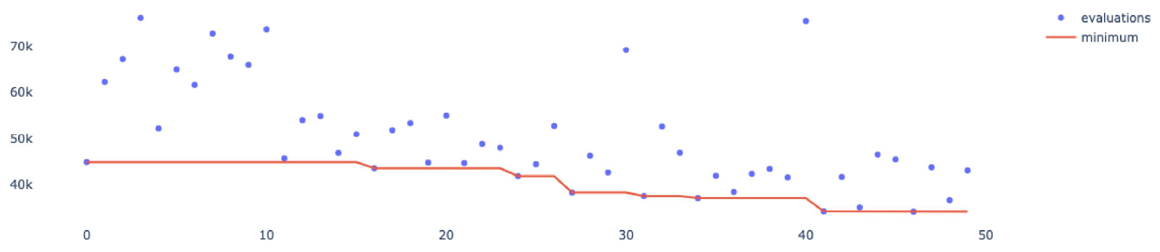
We interpret the remaining difference in the solution quality as random noise rather than a systematic advantage of one strategy over the other on this problem. All approaches seem robust enough to “solve” the use case. However, we want to stress that the performance can be quite application specific and may depend on the choice of tuning-parameters of the algorithm.

Progress Charts. In the following, we show a *progress chart* for each algorithm: Each simulation outcome is visualized by a blue dot in the respective graph. The red line shows the progression of the best objective value found so far during iteration.

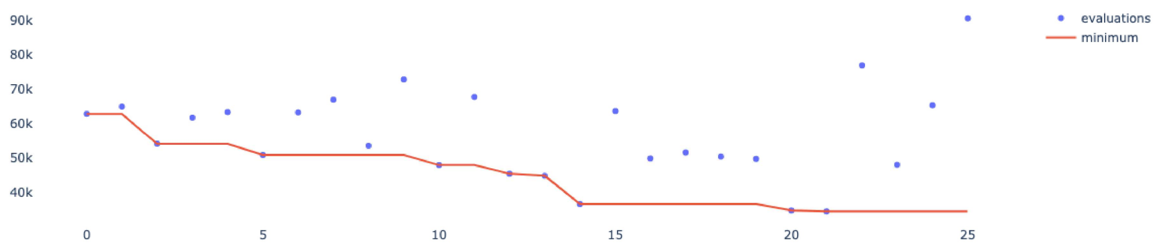
Stochastic Search. This algorithm acts as a baseline to compare the other algorithms with. 50 instances are chosen uniformly at random, simulated and evaluated. The approach ends with a target value of 43037 which is far away from the best known solution.



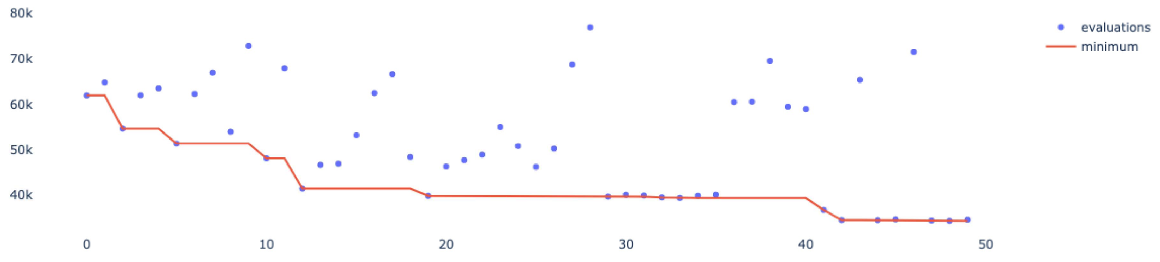
Evolutionary Algorithm. Five generations with population size ten. Tournament selection with one elitist individuum in the next generation. Discrete decisions are mutated uniformly at random with probability 0.1. Continuous decisions are mutated according to a normal distribution for which the mean is the current value and the variance is 0.1 times the difference between maximum and minimum possible value. Mutated decisions that lie outside the decisions domain are clipped to the next feasible value.



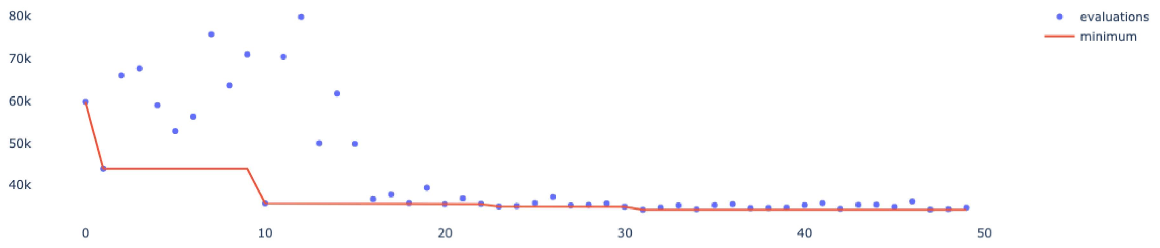
SMAC3 - Black Box Facade. Default parameters are applied.



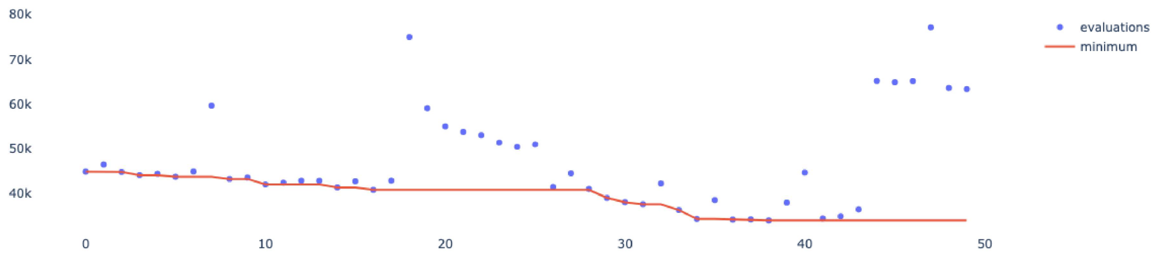
SMAC3 - Hyperparameter Optimization Facade. Default parameters are applied.



SMBO - Random Forest. The initial exploration phase uses 15 simulation runs using a Sobol-design. Each surrogate random forest is evaluated 10.000 times using a Sobol-design.



Stochastic Local Search. The probability to accept a worse solution as new incumbent is 0.1. A random restart is performed after 10 consecutive iterations that do not improve the best solution. The neighborhood is similar to the mutation of the evolutionary algorithm.



5 Conclusion

We proposed to use black-box optimization techniques on simulation models to identify good business decisions. This is useful in situations where building an optimization model is too time-consuming or costly, in particular when a simulation model already exists. To assess the quality of the approach, we conducted a small computational case study. We modeled the energy procurement of a mid-sized office building, making strategic decisions such as investment choices and simulating operations over time. The black-box routines were able to find near-optimal solutions demonstrating the efficiency of the approach. To the best of our knowledge, the method of automatically adding optimization capability to existing business decision simulations by using surrogate-based black-box optimization routines is new.

The obvious drackbacks of this approach are the lack of dual bounds on the cost function and the limited experimental experience. While the first aspect is inherent to the applied techniques, the experimental knowledge should be extended by adding more use-cases which we propose as further work on the topic.

Bibliography

[1] Audet, C., Hare, W.: Derivative-free and blackbox optimization. Springer (2017)

[2] Hutter, F., Kotthoff, L., Vanschoren, J.: Automated machine learning: methods, systems, challenges. Springer Nature (2019)

[3] Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Learning and Intelligent Optimization: 5th International Conference, Rome, Springer (2011)

[4] Gupta, S., Modgil, S., Bhattacharyya, S., Bose, I.: Artificial intelligence for decision support systems in the field of operations research: review and future scope of research. Annals of Operations Research (2022)

[5] Joe, S., Kuo, F.Y.: Constructing sobol sequences with better two-dimensional projections. SIAM Journal on Scientific Computing **30**(5) (2008)