Master's thesis
Master's Programme in Data Science

Automated Bayesian Chemical Reaction Optimization

Samuli Kinnunen

June 9, 2024

Supervisor(s): Assoc Prof Arto Klami, Prof Timo Repo

Examiner(s): Assoc Prof Arto Klami, Prof Timo Repo

University of Helsinki
Faculty of Science

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Degree programme
Faculty of Science	Master's Programme in Data Science

Tekijä — Författare — Author

Samuli Kinnunen

Työn nimi — Arbetets titel — Title

Automated Bayesian Chemical Reaction Optimization

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidantal — Number of pages
Master's thesis	June 9, 2024	60

Tiivistelmä — Referat — Abstract

Chemical reaction optimization is an iterative process that targets identifying reaction conditions that maximize reaction output, typically yield. The evolution of optimization techniques has progressed from intuitive approaches to simple heuristics, and more recently, to statistical methods such as Design of Experiments approach. Bayesian optimization, which iteratively updates beliefs about a response surface and suggests parameters both exploiting conditions near the known optima and exploring uncharted regions, has shown promising results by reducing the number of experiments needed for finding the optimum in various optimization tasks. In chemical reaction optimization, the method allows minimizing the number of experiments required for finding the optimal reaction conditions.

Automated tools like pipetting robots hold potential to accelerate optimization by executing multiple reactions concurrently. The integration of Bayesian optimization to automation reduces not only the workload and throughput but also optimization efficiency. However, adoption of these advanced techniques faces a barrier, as chemists often lack proficiency in machine learning and programming. To bridge this gap, *Automated Chemical Reaction Optimization Software (ACROS)* is introduced. This tool orchestrates an optimization loop: Bayesian optimization suggests reaction candidates, the parameters are translated into commands for a pipetting robot, the robot executes the operations, a chemist interprets the results, and data is fed back to the software for suggesting the next reaction candidates.

The optimization tool was evaluated empirically using a numerical test function, in a Direct Arylation reaction dataset, and in real-time optimization of Sonogashira and Suzuki coupling reactions. The findings demonstrate that Bayesian optimization efficiently identifies optimal conditions, outperforming Design of Experiments approach, particularly in optimizing discrete parameters in batch settings. Three acquisition functions; Expected Improvement, Log Expected Improvement and Upper Confidence Bound; were compared. It can be concluded that expected improvement-based methods are more robust, especially in batch settings with process constraints.

ACM Computing Classification System (CCS):
Computing methodologies → Artificial Intelligence → Knowledge and reasoning → Probabilistic reasoning
Applied computing → Physical sciences and engineering → Chemistry

Avainsanat — Nyckelord — Keywords

Bayesian optimization, chemical reaction optimization, robotics

Säilytyspaikka — Förvaringsställe — Where deposited

Kumpula campus, University of Helsinki

Muita tietoja — Övriga uppgifter — Additional information

Preface

This master's thesis encapsulates the research project that I conducted at the University of Helsinki at Catalysis and Green Chemistry (Catlab) research group and finalized at the Multi-source Probabilistic Inference (MUPI) research group as a part of Virtual Laboratories for Pharmaceutical project funded by Business Finland and participating companies.

I am grateful to my supervisors and examiners, Professor Timo Repo and Associate Professor Arto Klami, for their guidance and for enabling this highly interesting research project. I also want to thank Doctoral Researcher Joseph Install for his support with all the chemistry related aspects and for his assistance with the empirical research. Additionally, I am thankful to everyone at CatLab, who taught me enough chemistry to complete this project.

Contents

1 Introduction **1**

2 Chemical reaction optimization **5**

2.1 Background 5

2.2 Conventional optimization methods 6

2.2.1 One factor at the time 6

2.2.2 Design of Experiments 6

3 Bayesian optimization **11**

3.1 Problem setup 11

3.2 Gaussian process as a surrogate model 12

3.2.1 Gaussian process prior 13

3.2.2 Posterior distribution 13

3.2.3 Kernel 14

3.3 Acquisition function 17

3.3.1 Policies for selecting next experiment 17

3.3.2 Upper Confidence Bound 18

3.3.3 Expected improvement 19

3.4 Initialization of Bayesian optimization 19

3.5 Extensions of Bayesian optimization 20

3.5.1 Parallel Bayesian optimization 20

3.5.2 Dealing with mixed-valued variables 22

3.5.3 Process-constrained Bayesian optimization 24

3.6 Bayesian optimization in Ax and Botorch 25

4 Technical solution **27**

4.1 Development 27

4.2 Architecture 28

4.2.1 Overview 28

4.2.2 User interface module 29

4.2.3 Optimization module 30
4.2.4 OpenTrons control module 31
4.3 Code and availability 32

5 Empirical research **33**
5.1 Numerical test function 33
5.1.1 Data and methods 33
5.1.2 Results 34
5.2 Direct arylation reaction dataset 35
5.2.1 Data and methods 35
5.2.2 Results of unconstrained optimization 37
5.2.3 Results of process-constrained optimization 38
5.3 Real-time chemical reaction optimizations 39
5.3.1 Data and methods 39
5.3.2 Results 40

6 Conclusions **51**

Bibliography **55**

1. Introduction

The rise of artificial intelligence (AI) has gained a lot of attention in chemistry in recent years starting from prediction of chemical properties [23] and development of drug candidates to synthesis of desired compounds [1] and reaction optimization [21]. It is speculated that this rising interest will lead to the AI revolution in Chemistry [1]. In this thesis, the focus is in reaction optimization that typically takes place after the synthesis phase.

The purpose of reaction optimization is to find optimal reaction conditions, such as reaction time and temperature, to maximize the outcome of a reaction, which is often quantified using yield. Finding the optimal conditions may require from tens to more than a hundred experiments depending on the number of optimization parameters and their ranges. For example, the search space of the Direct arylation reaction that is optimized in Section 5.2 has 1728 parameter combinations. Since a single reaction can take hours and preparing it requires manual effort, there is a significant potential to save time and cost by reducing the number of experiments required for finding the optimum or by automatizing the process.

So far, solving reaction optimization problem has relied on chemist's intuition, simple optimization heuristics such as *one factor at the time (OFAT)* [41], and increasingly, on statistical optimization methods such as *Design of Experiments (DoE)*[54, 8]. The idea of DoE approach is to perform a set of experiments according to an experimental design, fit a first or second order model to the data and find the optimum by maximizing the model and running confirmation runs [54]. In recent years, *Bayesian optimization (BO)* has emerged as a promising approach for addressing the reaction optimization problem [46, 11] In BO, information and belief of an objective function or a reaction is accumulated by evaluating it (performing reactions) with different parameters (reaction conditions). By efficiently balancing exploration of unknown search space areas and exploitation of the areas near the current optima, BO can effectively identify optimal points to evaluate the function and eventually find the global maximum.

In addition to AI, also automation has potential to transform how chemists work. Automated tools such as flow reactors and pipetting robots have become increasingly popular during the past decade. Automation can smooth out solving the reaction opti-

mization task by improving the reproducibility and throughput of chemical reactions. Particularly, high-throughput experimentation has attracted considerable attention as it dramatically increases the number of reactions conducted per day to levels far beyond human capacity (exceeding 1000 reactions daily) [12].

So far, most efforts for automation have neglected decision making. Integrating feedback (e.g. analyzed product yield of a reaction) to the autonomous process has potential to transform automated experimentation to *autonomous optimization*, where also decision making is done independent of a chemist. This fundamentally changes the role of the chemist as the chemist is no longer in charge of the selection of experimental parameters and actions [12]. Combining automation and automatic decision making is in line with a broader vision of integrating physical processes, laboratory equipment, and AI tools to advance scientific research and development [29].

This thesis targets combining Bayesian optimization with automation to get closer to the stage, where the chemical reaction optimization is fully automatized. The main research questions of the thesis are:

1. How can BO be integrated with an automated pipetting robot to establish an autonomous chemical reaction optimization loop that minimizes the need for human intervention?

2. What is the performance of BO in chemical reaction optimization?

To answer the first research an automated optimization software was implemented in this study. The software consists of three modules: The *Optimization module*, the *OpenTrons control module* and the *User interface module*. The integration and information flow between the modules were managed using spreadsheets, allowing the modules to write and read data. The limitations of the robot were studied by performing tests where robot was controlled using the manufacturer's Python API. The physical limitation in the minimum pipetting volumes was addressed in the *OpenTrons control module* by developing a logic that dilutes additional stock solutions if the pipetting volumes get too low. The limitation of the robot heating all the reaction solutions in a batch to a single temperature was addressed in the *Optimization module*, where a process-constrained BO algorithm was implemented. To address the second research question, empirical research was conducted. The performance of BO was first validated in optimization of a numerical test function. Next, BO was tested in three different chemical reactions: Direct Arylation, Suzuki Coupling and Sonogashira Coupling. In the former, BO was also compared against widely used DoE approach.

Previous Bayesian optimization studies in reaction optimization have either neglected the automation [46] or used expensive high-end pipetting robot such as ChemSpeed with a price over 200k€ [11]. On the contrary, this study takes a step toward

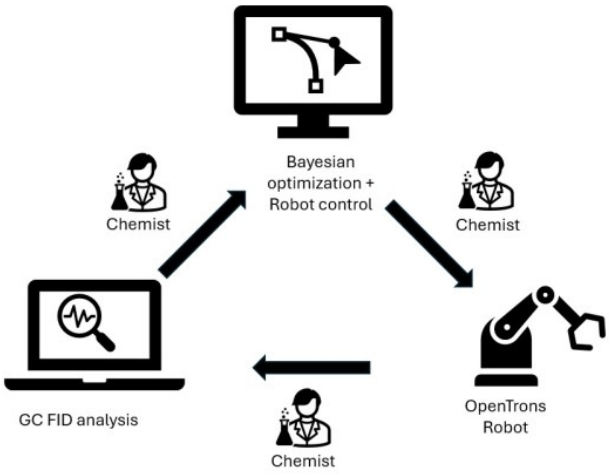

Figure 1.1: Semi-autonomous optimization loop. The main components are Automated Chemical Reaction Optimization Software (ACROS), OpenTrons pipetting robot, GC FID gas chromatography analysis equipment and a Chemist assisting in the process

democratizing automatized optimization by combining modern Bayesian optimization methods, graphical user interface and low-cost OpenTrons pipetting robot ($\sim$ 10k€) as shown in Figure 1.1. The framework requires that human is in the loop for supporting in analyzing of the results and setting up the robot for each batch. The main contributions of the thesis are:

1. To the best of the author's knowledge, the implemented system is the first attempt to develop easy-to-use automatized optimization tool. The software offers graphical user interface, which can be seen as an important factor behind the popularity of current design of experiment software such as MODDE [24].

2. This study demonstrates the strong performance of implemented Bayesian optimization using a numerical test function, pre-collected chemical reaction data and optimizing reactions in real-time. BO is compared to the DoE approach and it is shown that BO outperforms DoE in optimization of discrete variables of Direct arylation reaction. It is also empirically shown that process-constrained batch BO, where temperature is constant for each batch, can efficiently find optimal reaction conditions.

The thesis first introduces the reader to the theoretical background of reaction optimization in Chapter 2 and Bayesian optimization in Chapter 3. Next, the development and architecture of technical solution is described in Chapter 4. Chapter 5 studies the empirical performance of the implemented optimization framework. Finally, the results are concluded in Chapter 6.

2. Chemical reaction optimization

This chapter explores chemical reaction optimization from the optimization perspective, without delving deeply into the chemistry. The chapter begins with the background and problem setup outlined in Section 2.1. Subsequently, Section 2.2 provides an overview of two commonly employed optimization methods.

2.1 Background

Chemical reaction optimization aims to uncover the optimal reaction parameters that maximize the desired reaction outcome, often measured in terms of product yield. Prior to exploring various methods for optimizing reactions, it is important to define some fundamental terms:

Factors, independent variables or *reaction conditions*, denoted as c, are experimental variables that can be changed independently of each other. A typical set of optimization parameters include continuous parameters such as temperature, concentrations and time, and categorical parameters such as catalyst or solvent selection [11].

Response or *dependent variable*, denoted as r, refers to a measured result of an experiment. In reaction optimization, a typical metric to quantify results is the reaction yield percentage $y \in [0, 1]$, that can be defined as a ratio of the actual weight or molar amount of an intended product a and the theoretical weight or molar amount of intended product t of a chemical reaction as defined here [57]:

$$y = \frac{a}{t} \times 100\% \tag{2.1}$$

Also other metrics can be used such as selectivity that measures the amount of desired reaction product with respect to total amount of all products, or conversion that refers to the amount of reactants that are converted to products [40].

In summary, a reaction can be viewed as a system taking reaction conditions $c \in \mathbb{R}^d$, as an input and yielding a quantifiable product as a response. d stands for the number of reaction conditions to be optimized. Thus, reaction can be formalized by a function

$$r = R(c), \tag{2.2}$$

where reaction function R: $\mathbb{R}^d \rightarrow \mathbb{R}$ determines how reaction conditions affect to the response r[61].

Evaluating reaction function R is typically possible only by performing an experiment, where reaction is performed in one or more in reaction conditions. The task of chemical reaction optimization is to find reaction conditions that maximize the response: $\arg\max_c R(c)$. Since R lacks mathematical formulation that would allow solving the optimization problem, various methods for chemical reaction optimization have been developed.

Traditionally, optimization in this context has relied on intuition or OFAT method [53]. However, the rise of computers and user-friendly software packages have increased the popularity of DoE. More recently, research and development of reaction optimization methods has shifted towards leveraging machine learning based strategies such as Reinforcement Learning (RL) [61], Stable Noisy Optimization by Branch and Fit (SNOBFIT) [38], and Bayesian Optimization (BO) [12][46]. The latter will receive detailed coverage in Chapter 3.

2.2 Conventional optimization methods

2.2.1 One factor at the time

Possibly the simplest and most commonly used systematic approach for reaction optimization is one factor at the time optimization, where a single parameter c_i is varied, while others are kept fixed. This is done iteratively for each c_i. The method neglects the interaction effect between reaction conditions and can lead to sub-optimal results [41]. Figure 2.1 illustrates how OFAT can lead to a sub-optimal result.

2.2.2 Design of Experiments

Design of Experiments (DoE) is a set of experiments that allows modelling the response surface between independent variables and the response. DoE is sometimes referred more widely as an optimization technique [54], statistical tool [13], or framework [26]. In this thesis, *experimental design* refers to the set of experiments and *DoE approach* refers more widely to the optimization process that can be summarized to four phases:

1. Designing an experiment campaign according to the chosen experimental design
2. Performing experiments
3. Building and optimizing response surface model

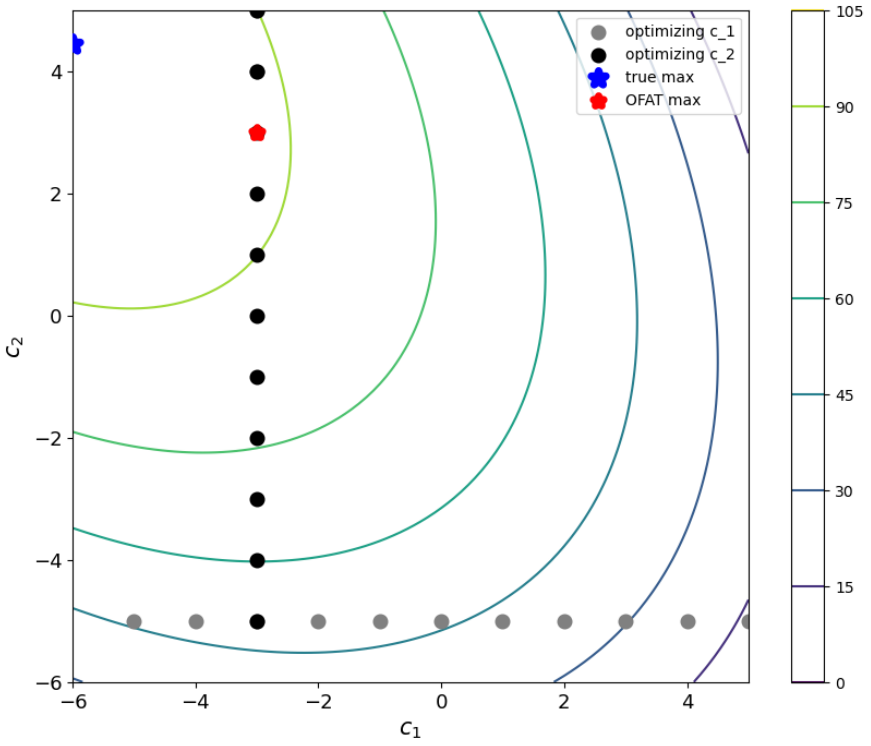

Figure 2.1: Illustration of one factor at the time optimization of black-box function f leading to a sub-optimal result. Varying first factor c_1 and keeping c_2 constant to find maximum of $c_1(-3)$, and then keeping c_1 constant at its maximum value and varying parameter c_2 to find its maximum ($x_2 = 3$) fails finding the global maximum. The result of OFAT optimization is $f(-3, 3) = 93$ while the true maximum is at $f(-5, 4) = 100$.

4. Running confirmation runs.

Designing an experiment campaign typically involves conducting initial experiments aimed at identifying the most influential parameters for optimization and establishing their respective ranges [26]. Once the parameter space is defined, an experimental design determines the experiments required for constructing the response surface model.

Second-order regression model is typically used in modelling the response surface in relation to the experimental factors. Fitting a quadratic model requires experiments from at least three levels of each factors [8]. For example, a three-level design for exploring the effect of reaction temperature (10-50 $C°$) would use the levels -1, 0, and 1 corresponding to 10, 30, and 50 $C°$, respectively. The number of experiments required depends on the factorial design. Full factorial design, illustrated in Figure 2.2 requires $N = 3^k$ experiments, k denoting number of factors. As the number of factors increases, the number of experiments grows exponentially. To address this challenge, designs requiring fewer experiments have been developed. For instance, a class of experimental designs called Fractional designs reduce the number of experiments compared to Full factorial design [8].

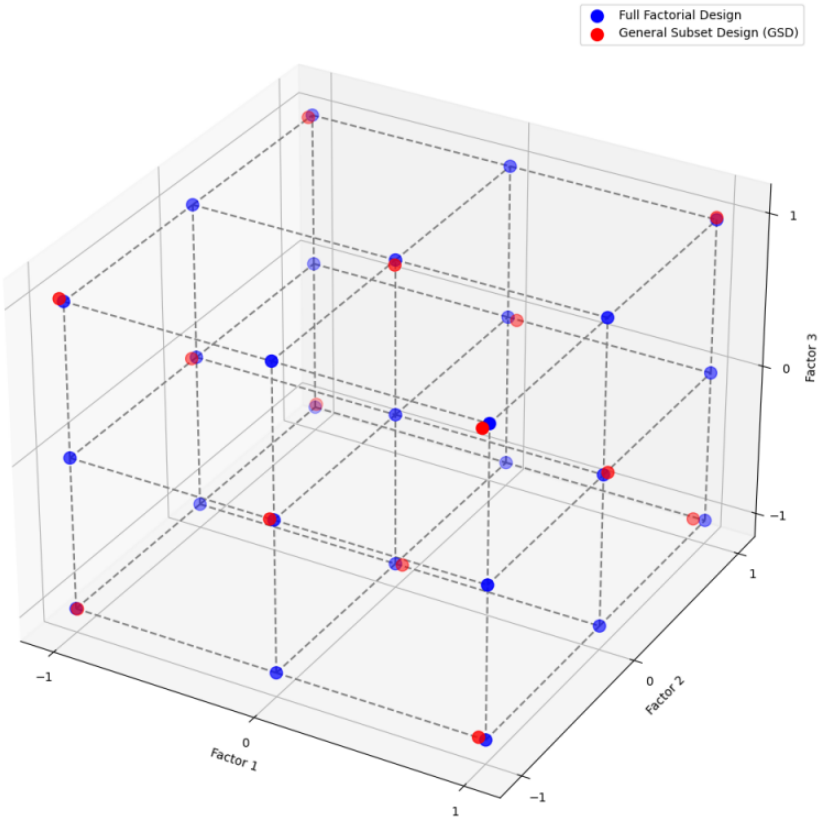

Figure 2.2: Illustration of Full factorial design and General subset design with three factors and levels. Random noise is added to General subset design points to visualize the overlapping points.

Generalized Subset Design (GSD) [52] is an example of an experimental design, that can accommodate both continuous and categorical factors across multiple levels. GSD offers a balanced subspace where each factor level is evaluated an equal number of times, and each factor can be assessed independently of others. Essentially, GSD comprises reduced subsets of the Full factorial design, each complementing the others. This characteristic enables the design to be tailored to fit within a smaller experimental budget. This also means, that GSD can be complemented with additional subsets to address any lack of fit in the response surface derived from the initial experiments. GSD is illustrated in Figure 2.2. After the experiments have been performed according to the experimental design, preferably in randomized order, the chosen model can be fitted. A quadratic model to predict the value of response variable $\hat{r}$ in relation to n reaction conditions c can be written as

$$\hat{r} = b_0 + \sum_{i=1}^{n} b_i c_i + \sum_{i=1}^{n} b_{ii} c_i^2 + \sum_{1<=i<=j}^{n} b_{ij} c_i c_j + \epsilon, \tag{2.3}$$

where b_0 stands for an intercept term, b_i corresponds to a coefficient term for linear terms, b_{ii} for quadratic terms, and b_{ij} for interaction terms. In matrix notation, the

model can be written as

$$\mathbf{r} = \mathbf{X}\mathbf{b} + \boldsymbol{\epsilon}, \tag{2.4}$$

where $\mathbf{r}$ is the vector of observed responses, $\mathbf{X}$ is the design matrix, which includes the intercept, linear, quadratic, and interaction terms and $\mathbf{b}$ represents the vector of coefficients b_0, b_i, b_{ii}, b_{ij}. To fit this model, Ordinary Least Squares method (OLS) can be used. In OLS, the target is to minimize the sum of squared differences between the observed responses r and predicted responses $\hat{r}$. Mathematically, this can be expressed as

$$\arg\min_{\mathbf{b}} ||\mathbf{r} - \mathbf{X}\mathbf{b}||^2 \tag{2.5}$$

Solving the minimization problem results in:

$$\mathbf{b} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{r}. \tag{2.6}$$

The response surface allows visual inspection and calculation of critical points (a local minimum, maximum or saddle point). To find the critical points, partial derivatives of r with respect to each c_i should be calculated and set them to zero:

$$\frac{\partial \hat{r}}{\partial c_i} = b_i + 2b_{ii}c_i + \sum_{j \neq i} b_{ij}c_j = 0 \quad \text{for all } i. \tag{2.7}$$

The critical points can be obtained by solving the system of equation of the partial derivatives. Next, a Hessian matrix H, which is a square matrix of second order derivatives, is calculated:

$$H = \begin{bmatrix} \frac{\partial^2 \hat{r}}{\partial c_1^2} & \frac{\partial^2 \hat{r}}{\partial c_1 \partial c_2} & \cdots & \frac{\partial^2 \hat{r}}{\partial c_1 \partial c_n} \\ \frac{\partial^2 \hat{r}}{\partial c_2 \partial c_1} & \frac{\partial^2 \hat{r}}{\partial c_2^2} & \cdots & \frac{\partial^2 \hat{r}}{\partial c_2 \partial c_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 \hat{r}}{\partial c_n \partial c_1} & \frac{\partial^2 \hat{r}}{\partial c_n \partial c_2} & \cdots & \frac{\partial^2 \hat{r}}{\partial c_n^2} \end{bmatrix}. \tag{2.8}$$

Based on the following rules, the nature of critical points is determined:

- If H is positive definite at a critical point, the function has a local minimum there.
- If H is negative definite at a critical point„ the function has a local maximum there.
- If H is indefinite, the critical point is a saddle point (neither maximum nor minimum).

Lastly, also the boundaries of the search space are evaluated to identify the global optimum.

The last phase of the DoE approach discussed here is confirmation. While there are different meanings for confirmation, typically confirmation runs mean making additional experiments in the region of the optimum. It is suggested that in the case of five factors, 5-10 confirmation experiments should be performed [51]. The need for confirmation arises from the need to validate the consistency of the results. Experimental results can have stochastic noise that also causes variability in the response surface and estimated optimum [26]. Confirmation runs may also be required when the reaction is scaled up or when regulatory requirements demand that the process factors are actually tested and not based only on an estimated response [51].

3. Bayesian optimization

This chapter first covers the problem setup of BO in Section 3.1 and then explains the main components in Sections 3.2 and 3.3. The initialization of Bayesian optimization is covered in Section 3.4. Extensions of BO; batch optimization, process-constrained optimization, and mixed-type variables optimization; are covered in Section 3.5 after explaining the basic version.

3.1 Problem setup

Many optimization problems are solved using gradient-based optimization methods, where gradient of the objective function is utilized to iteratively approach the optimum. The need for BO arises from the optimization problems, where there is not closed-form formulation of the objective function allowing computation of the gradient. This means that the objective function f can be only observed through point-wise observations. Often it is assumed that the observations are noisy meaning that the measured response of a chemical reaction may include uncertainty related to the measurement process. For example, analysis test sample size, homogeneity and measurement device as well the chemistry itself can cause such noise [32]. Because the evaluations of the objective function is typically time-consuming and evaluations are expensive, exhaustive search through the search space is not feasible. The target of BO is to efficiently solve these expensive-to-evaluate black box optimization problems [14].

Mathematically, the objective of BO is to discover a parameter combination x^* that maximizes the black box function f:

$$x^* = \arg\max_{x \in \chi} f(x), \tag{3.1}$$

where x represents the parameters of the unknown function $f(x)$. The parameter x is confined to the search space χ, which is typically a compact subset of $\mathbb{R}^d$. In most successful applications of BO, the number of dimensions d is less than 20 [17].

BO has two main components: a *surrogate model* for approximating the objective function and an *acquisition function* for determining the value of evaluating objective function at a new candidate point x. Equation 3.1 can be optimized using a BO

algorithm. The fewer evaluations the algorithm requires for finding the optimal x* (or sometimes close to optimal), the better the algorithm works. As explained in Algorithm 1, at each iteration i, $f(x_i)$ is evaluated. After the function is evaluated, the algorithm updates the surrogate model. Next, a candidate point x_{i+1} that maximizes the *acquisition function* is selected for the next iteration. The optimization continues as long as $maxIterations$ has been reached or other stopping criterion is met. The stopping criteria can be, for example, an objective threshold which need to be exceeded.

Algorithm 1 Pseudo-code for Bayesian optimization algorithm

1: Initialize: Choose initial x_0, set iteration counter $i = 0$, define $maxIterations$ and $stoppingCriterion$
2: **while** $i <= maxIterations$ and $stoppingCriterion$ not met **do**
3: Evaluate $f(x_i)$ to obtain y_i
4: Update the surrogate model with all available data $\{(x_i, y_i)\}_{i=1}^{n}$
5: Select candidate point x_{i+1} that maximizes the acquisition function
6: $i \leftarrow i + 1$ ▷ Move to the next iteration
7: **end while**

3.2 Gaussian process as a surrogate model

A surrogate model has two main purposes. First, the surrogate model approximates the true function based on the observations providing understanding about the relationship between input variables and an observed objective function. Second, the surrogate model provides understanding about the uncertainty of the current approximation. This uncertainty can be leveraged in finding the next candidate points to evaluate.

The surrogate model typically belongs to the family of Bayesian models due to their ability to quantify uncertainty. The Bayesian regression model called *Gaussian process (GP)* regression has become a standard choice for a surrogate model [17]. While GP is the most commonly used surrogate model, also other models such as deep GPs [22], random forest [25] and Bayesian neural networks [20] have been proposed. Still, many recent methodological advances such as new acquisition models [4] [5], and advances in applications of Bayesian optimization such as Bayesian reaction optimization [11] [46] have built their solutions on top of GP surrogates. Thus, GP is presented this section.

3.2.1 Gaussian process prior

A Gaussian prior is set on the function $f(x)$. Assuming independent identically distributed normally distributed noise ϵ_n, the examined function becomes $y_n = f(x) + \epsilon_n$ [60]. According to the prior, values of this function are multivariate normally distributed with a constant, typically zero mean, and with a covariance matrix defining the covariance between different covariates of $f(x)$. In practice, the covariance matrix is determined by a *kernel function* that assigns covariance based on the distance between evaluated points. The multivariate normal prior distribution that follows is

$$y|f, X \sim \mathcal{N}(0, K(X, X) + \sigma_n I) \tag{3.2}$$

where $K(X, X)$ is $n \times n$ covariance matrix consisting of the covariances evaluated at all pairs of input points. To address noisy observations, an observation noise σ_n is added to the diagonal of the covariance matrix.

Typically, the interest is on how to incorporate the knowledge of the training data to the prior. The joint distribution of the training observations y and function values f_* at some new point according to the prior can be written as

$$\begin{bmatrix} y \\ f_* \end{bmatrix} = \mathcal{N}\left(0, \begin{bmatrix} K(X, X) + \sigma_n I) & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right), \tag{3.3}$$

where X refers to the locations of the observations and X_* to the test locations. Constructing covariance matrices $K(X, X)$, $K(X_*, X)$, $K(X, X_*)$ and $K(X_*, X_*)$ is discussed in more detail in Section 3.2.3.

3.2.2 Posterior distribution

Posterior distribution can be loosely interpreted as a summary of prior distribution and observations. The process of updating prior belief with observations is illustrated in Figure 3.1. An analytical form of a posterior distribution is discussed in this section.

Given observations $\{(x_i, y_i)\}_{i=1}^n$, to get a posterior probability distribution for an arbitrary test point, the prior probability distribution is conditioned on the observations using Bayes' rule. The resulting posterior distribution is

$$f_*|X, y, X_* \sim \mathcal{N}(\bar{f}_*, \mathrm{cov}(f_*)) \tag{3.4}$$

with the mean and covariance functions

$$\begin{aligned} \bar{f}_* &= K(X_*, X)[K(X, X) + \sigma^2 I]^{-1} y \\ \mathrm{cov}(f_*) &= K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma^2 I]^{-1} K(X, X_*), \end{aligned} \tag{3.5}$$

where the GP posterior mean $\bar{f}_*$ is determined by the prior and observations y that are weighted by the kernel [60]. Posterior covariance matrix is determined by inputs of observations and test set, and the kernel function. The first element $K(X_*, X_*)$ represents the prior variance that is subtracted a positive term that represents the information gained from the observations. With most kernels, the closer the evaluated point is to the observations, the closer the mean is to the observations and the smaller is the variance. Furthermore, the kernel has an important effect both on the shape of predicted function and its variance.

If also observation noise ϵ follows a normal distribution, GP regression is conjugate resulting in that posterior and marginal likelihood is obtained in closed form [3]. GPs can be also used with different likelihood functions i.e. in classification by setting likelihood function to Bernoulli distribution:

$$y|f(x) \sim \text{Bernoulli}(\sigma(f(x))), \tag{3.6}$$

where GP prior is used for functions f. Changing the likelihood function breaks the conjugacy, so approximate inference such as Laplace approximation or Markov Chain Monte Carlo sampling should be used [60].

3.2.3 Kernel

A kernel describes the covariance of GP random variables. A valid kernel needs to be positive definite and symmetric [60]. It is common to use stationary kernels which output depends on the distance $r = x - x'$ between the inputs [35].

Radial basis function (RBF) kernel is an example of a typical and simple stationary kernel. It can be defined as

$$k(x, x') = e^{\frac{-||x-x'||^2}{2l^2}}, \tag{3.7}$$

where the length scale l determines how quickly the correlation between function values decays as the input points move apart from each other. A shorter length scale implies rapid fluctuations in the function values, while a longer length scale suggests smoother variations over greater distances. For example, keeping the distance between inputs constant and changing the length parameter from 1 to 0.3 decreases the correlation from 0.61 close to zero: Let the distance between the inputs be $||x - x'|| = 1$ and the length parameter be $l = 1$:

$$k(x, x') = e^{\frac{-1}{2*1^2}} \approx 0.61,$$

and similarly, let the distance between inputs be $||x-x'|| = 1$ and the length parameter be $l = 0.3$:

$$k(x, x') = e^{\frac{-1}{2*0.3^2}} < 0.01$$

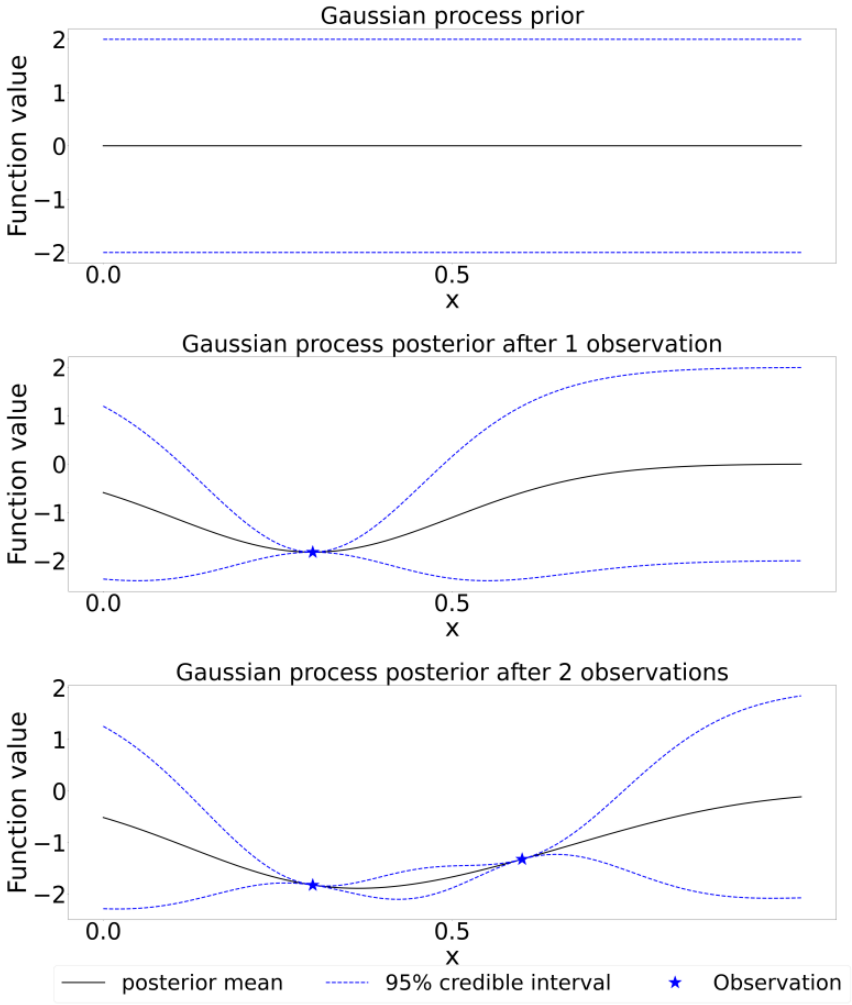

Figure 3.1: Illustration of updating observations to GP over 0-2 time steps.

As defined earlier, the joint distribution and the posterior distribution require defining covariance matrices $K(X, X)$, $K(X_*, X)$, $K(X, X_*)$ and $K(X_*, X_*)$. A covariance matrix can be defined as:

$$K(X', X) = \begin{bmatrix} k(x'_1, x_1) & k(x'_1, x_2) & \cdots & k(x'_1, x_n) \\ k(x'_2, x_1) & k(x'_2, x_2) & \cdots & k(x'_2, x_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(x'_m, x_1) & k(x'_m, x_2) & \cdots & k(x'_m, x_n) \end{bmatrix},$$

where $X = \{x_1, x_2, \ldots, x_n\}$ and $X' = \{x'_1, x'_2, \ldots, x'_m\}$ are sets of data points and the kernel function is denoted by $k(x, x')$.

Since the RBF kernel tends to smooth out sharp jumps in the observations, sometimes a kernel that better captures "rougher" functions is needed. In many application, it is beneficial to use *Matérn kernel* that captures local "wiggles" without a need of tuning length hyperparameter very small [35]. The Matérn kernel is often used with hyperparameter ν being $p + 1/2$, where p is a non-negative integer. Hence, the kernel

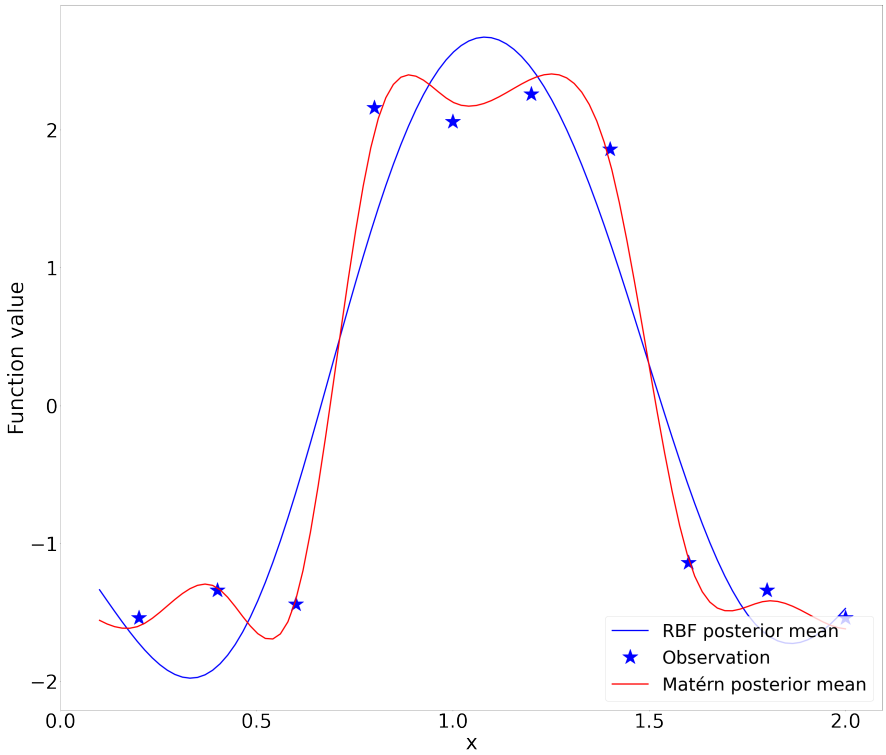

Figure 3.2: Comparison of GP posterior means using RBF and Matérn kernels. For the both kernels, the length parameter is $l = 0.2$ and observation noise is $sigma = 0.1$.

takes a relatively simple form. Often used parametrizations of Matérn kernel, that have a simplified form, are Matérn with $\nu = 3/2$ and $\nu = 5/2$:

$$k_{\nu=3/2} = \left(1 + \frac{\sqrt{3}r}{l}\right)\exp\left(-\frac{\sqrt{3}r}{l}\right) \tag{3.8}$$

$$k_{\nu=5/2} = \left(1 + \frac{\sqrt{5}r}{l} + \frac{5r^2}{3l^2}\right)\exp\left(-\frac{\sqrt{5}r}{l}\right) \tag{3.9}$$

which similarly to the RBF, have length l as a parameter. Figure 3.2 shows an example of how RBF kernel smooths out the sharp jump in observations more than the Matérn kernel. Kernel parameters, such as l, significantly influence the function by determining the correlation between the outputs. Optimizing kernel parameters such as l would be possible by exhaustively search over a grid of l, with validation loss as an objective. However this approach is slow [35]. A faster way is to maximize marginal likelihood with respect to the parameter. To do this, function f is first integrated from the likelihood of the parameters θ:

$$p(y|X,\theta) = p(D|\theta) = \int p(y|fX,\theta)p(f|X,\theta)\,df_X \tag{3.10}$$

where $p(D|\theta)$ is often called as the marginal likelihood and θ denote hyper parameters. Since, f is a GP, it is possible to compute the integral using the marginal likelihood of

the corresponding Gaussian. This leads to

$$\begin{aligned}\log p(D|\theta) = -\frac{1}{2}(y-\mu_X)^T(K_{X,X}+\sigma^2 I)_{\sigma}^{-1}(y-\mu_X) \\ -\frac{1}{2} * \log|K_{X,X}+\sigma^2 I| - \frac{N}{2}\log(2\pi)\end{aligned} \tag{3.11}$$

where $y-\mu_X$ is the distance between observations and predicted values, $\log|K_{X,X}+\sigma^2 I|$ is the log determinant of the kernel and the last term $\frac{N}{2}log(2\pi)$ is just a constant. Maximizing this marginal likelihood allows optimization of the kernel parameters. A simple way would be again to exhaustively search over a grid, but this approach can be slow depending on the kernel. For more complex kernels, parameters could be optimized by calculating a gradient of the marginal likelihood and using a gradient-based optimizer.

3.3 Acquisition function

At each iteration of the BO algorithm 1, a new candidate point is selected. The selection of the candidate is done by maximizing an acquisition function that leverages a surrogate model to guide the selection. The acquisition function assesses the expected utility of evaluating an objective function [45]. As the overall target is to find the global optimum, unexplored regions of the surrogate model or the areas close the current optimum are typically considered having high expected utility. The next points to evaluate are determined by maximizing this expected utility.

This section first covers the general form of an acquisition function and typical policies in section 3.3.1. Second, two acquisition functions, *Upper Confidence Bound (UCB)* and *Expected Improvement (EI)* are explained in more detail in Sections 3.3.2 and 3.3.3. These are the same acquisition functions that are used in chemical reaction optimization in Chapter 5.

3.3.1 Policies for selecting next experiment

Acquisition function assesses the areas of the search space utilizing both mean and uncertainty of the GP surrogate. Exploitation means seeking points in areas, where the model predicts a high value. Exploration, in turn, means focusing on the areas, where the uncertainty is high. The choice between exploitation of high expected performance and exploration of undiscovered areas is often called "exploration vs. exploitation trade-off" [17]. Acquisition functions provide tools to balance between these aspects. In practice, this means that the algorithm tries to reduce uncertainty in the areas, where the global optimum might still be located. Sometimes this means exploring completely

unexplored areas, while other times it involves investigating areas near points that have previously resulted in good values but still have some uncertainty.

Several policies for selecting the next experiments have been proposed [45]. Firstly, improvement-based acquisition functions such as *Probability of Improvement (PI)* [31], *Expected Improvement (EI)* [27], and *Knowledge Gradient (KG)* [16] prioritize points likely to enhance the target either from the current maximum of observations (PI and EI) or from the current maximum mean of the surrogate model (KG). Secondly, optimistic policies like *Upper Confidence Bound (UCB)* [50] favor points with high mean and uncertainty of a surrogate model in order to find location that could result in high values. Lastly, there are information-based policies such as *Thompson Sampling (TS)* [55], that randomly samples points from the surrogate model and selects the point with the highest utility.

Acquisition functions can be generalized to a form:

$$\alpha(x; \phi, D) = \mathbb{E}[\alpha(g(f(x)), \phi)|D] \tag{3.12}$$

where ϕ are parameters independent of x, D is data consisting of observations of $f(x)$ and g is an objective function. Here α is a utility function which represents the acquisition function [5]. In practice, finding the most optimal candidate x means taking expectation over $f_D(x)$. This expectation can be calculated analytically only for some of the acquisition functions in a limited setting e.g. non-parallel Expected Improvement [5]. When an analytic solution is not available, approximations like Monte Carlo integration may be employed [5].

3.3.2 Upper Confidence Bound

The upper confidence bound criterion is a common way to balance between exploration of the regions in the search space that have a lot of uncertainty, and exploiting areas near the current maximum. UCB determines the next candidate point by selecting a point with the highest upper bound value of a GP surrogate. GP-UCB is computed as

$$\alpha_{UCB}(x; D_n) = \mu_n(x) + \beta_n \sigma_n(x), \tag{3.13}$$

where μ_n is the mean of the posterior and σ_n is the diagonal of the posterior covariance matrix for the corresponding x. High μ_n can be seen to favor exploitation of points near the current maximum whereas high σ_n to favors exploration of points with a lot of uncertainty. This trade-off can be regulated by adjusting hyperparameter β_n. Any quantile of the posterior can be used as an upper bound by using corresponding β_n. Figure 3.3 demonstrates the acquisition function using $\beta = 2$ which corresponds to about 95% credible interval.

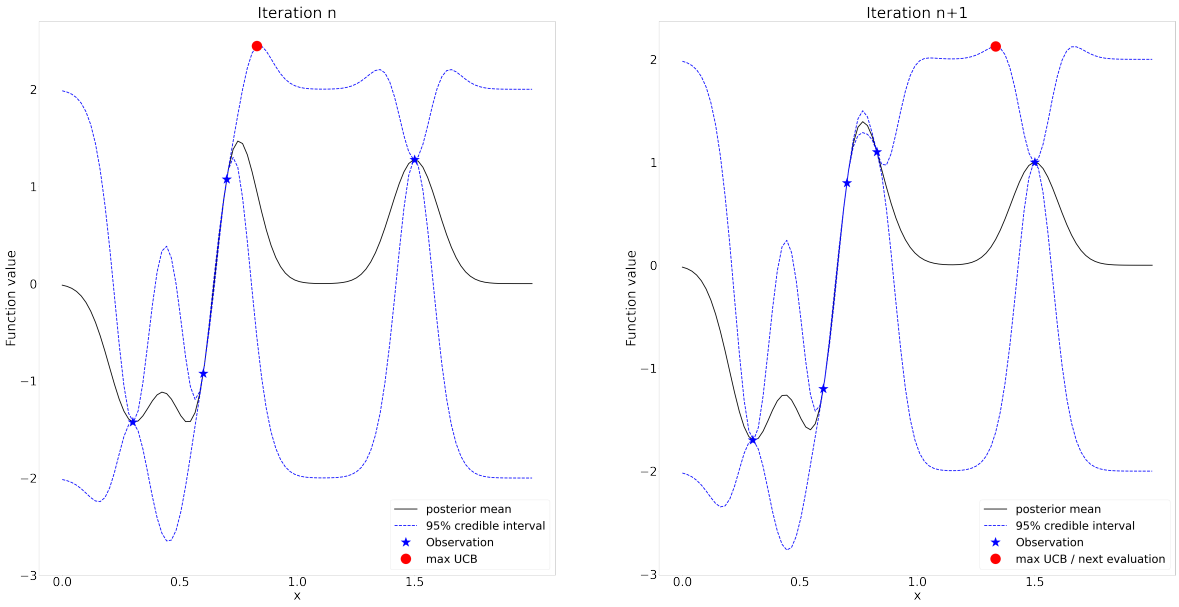

Figure 3.3: Illustration of BO using UCB as an acquisition function with the hyperparameter $\beta_n = 2$. When the point with the highest UCB value in the left plot is evaluated, the uncertainty in that area reduces and the location of the highest UCB value shifts in the right plot

3.3.3 Expected improvement

Expected improvement (EI) is a popular and powerful acquisition function that has shown solid performance both in optimization of numerical functions [45], and in applied optimization problems such as reaction optimization [46]. EI can be defined as

$$EI_n(x) := E_n[[f(x) - y_n^*]^+ | D] \tag{3.14}$$

where $[\cdot]^+$ denotes the $max(0, \cdot)$ operation. Hence, the equation stands for an expectation of the difference between the value of the current best evaluated point y_n^* and yet unknown observation $f(x)$ taken under the posterior distribution given evaluations $D = \{x_n, y_n\}_{n=1}^N$. Calculation of EI is more challenging, and it is not covered here, but can be found, for example, in [27].

3.4 Initialization of Bayesian optimization

BO typically requires having some initial data [42]. This data allows fitting the surrogate model and maximizing the acquisition function to locate the next points to evaluate. Without the initial data, a GP surrogate model would rely solely on its prior, resulting in a constant mean and uncertainty as illustrated in Figure 3.1. Consequently, the acquisition function values would be constant across the search space, and determining the next experiment(s) would need to be based on something else that

maximizing the acquisition function.

In practice, the initial data can consist of experiments that have been performed to define the search space χ. For example, in chemical reaction optimization, it is typical to perform a set reactions to screen the reaction conditions that have the largest effect on the response [8]. However, in some cases there might not be any data available for the initialization. In this case, the initial points to evaluate can be selected randomly, using knowledge from previous similar experiments [15], using *Latin hypercube sampling (LHS)* [33] or using *Sobol sequences* [48]. Sampling methods such as LHS and Sobol sequences are typically preferred over random sampling to efficiently explore the search space [42]. They generate samples that are more evenly spaced around the search space. Such sequences are called Low Discrepancy Sequences (LDS). Discrepancy is a measure of deviation from uniformity. Figure 3.4 illustrates sampling in two dimensional space using random sampling, LHS and Sobol sequences.

In a two-dimensional space, LHS divides each dimension into N equal intervals, resulting in a grid of N^2 squares known as *Latin squares*. For both dimensions, one point is sampled from each interval forming sequences of N points $x_1, ..., x_N$ and $y_1, ..., y_N$. These sequences are then paired randomly to populate the two-dimensional space. This method ensures that each row and each column contains exactly one sample point. The method can be generalized to higher dimensional spaces, where the k dimensional space is divided into N^k hypercubes forming a *Latin hypercube*.

A sobol sequence is constructed of a deterministic sequence of samples converging quickly towards a uniform distribution [42]. Sobol sequences target meeting three main properties [30]:

1. Maximal uniformity of distribution as $N \to \infty$
2. A good distribution also for smaller number of samples
3. Computational efficiency

It has been has demonstrated that Sobol sequences outperform LHS in applications such as numerical integration [43] and in financial engineering [30]. As there is little evidence of the performance in BO setup, Sobol is used in this study. A lengthy and detailed mathematical explanation to generate Sobol sequences can be found in [10].

3.5 Extensions of Bayesian optimization

3.5.1 Parallel Bayesian optimization

The sequential version of BO as described in the previous sections updates the surrogate model after each evaluation. Consequently, the next point is selected based on the

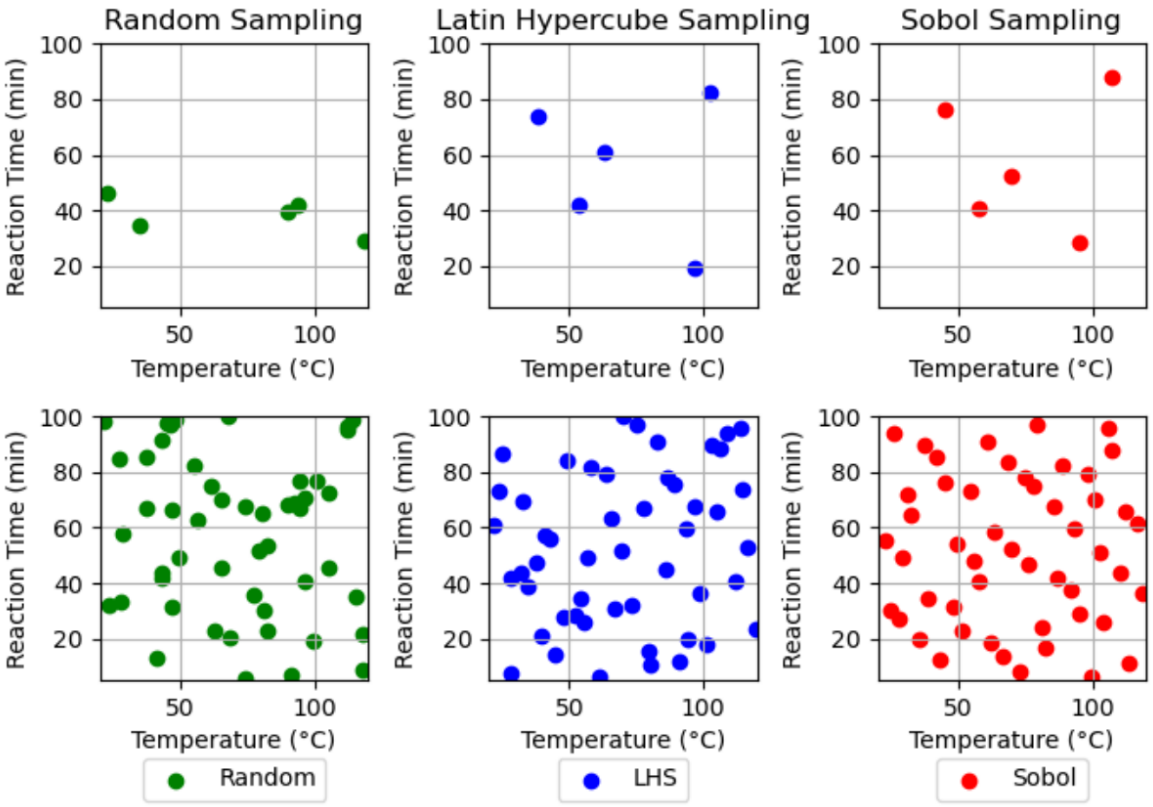

Figure 3.4: Illustration of Random, LHS and Sobol samples. 5 points are sampled using each sampling method in the top row and 50 points in the bottom row.

updated surrogate. It is assumed that the previous experiment results are in use when generating inputs for the next experiment [45]. However, in many applications, it is more affordable and faster to perform experiments or evaluations in parallel. This applies especially well to reaction optimization, where running a single reaction can take hours or days to run to completion [46] and reactions can typically be performed in parallel. At least two ways to address parallel BO have been presented: turning sequential candidate selection into parallel candidate selection [45] and modifying the acquisition function for batch setting [58]. These approaches are covered next.

Sequential greedy strategies adapt sequential candidate selection to parallel setting. The first point is generated the same way as in a standard BO. To generate the next point, the surrogate model is updated with an estimate of the observation value based on the current surrogate function. The surrogate is updated with the estimated value before generating the next candidate for the batch. This procedure is repeated until the whole batch is generated [4]. An example such of a strategy is the Kriging believer that uses the predictive mean of a surrogate function as the value to estimate the not-yet performed experiments [45]. Another strategy is to sample a set of values from the posterior for each unfinished experiments and taking empirical mean of these

samples [47]. Although this approach is simple, it is also somewhat naive as it only considers a single future scenario when selecting the next points.

A family of acquisition functions called *Multi-points expected improvement (qEI)* solves the same parallel candidate generation problem by modifying the acquisition function instead of generating points one by one. qEI has been shown to find optimal solution faster than sequential greedy strategies especially when the batch size q increases [58]. The qEI acquisition function computes directly the expectation for the whole candidate set of points X. This means that there is no need of generating candidates one by one [58]. qEI is given by

$$qEI(X) = E_n\left[max_{i=1,\dots,q}\left(f(x_i) - y_n^*\right)^+\right] \tag{3.15}$$

where y_n^* is the value of the current best evaluated point and $max_{i=1,\dots,q}(f(x_i))$ stands for the best value of the evaluation of q new points. Consequently, $E_n[\cdot]$ signifies the expected improvement of the candidate set, computed with respect to the posterior distribution of the Gaussian surrogate. Maximizing this expectation over different candidate set of points can be written as

$$\arg\max_{X \in H} qEI(X) \tag{3.16}$$

where H limits the evaluated batch of points to have at least distance $r > 0$ from each other. This optimization problem can be solved using a multi-start stochastic gradient ascent framework [58] which samples from the posterior to calculate the gradient of qEI.

qEI outperforms the sequential greedy strategy [58]. However, expected improvement suffers from the problem of vanishing values and gradients especially in higher dimensions and with more collected data [4]. The logarithmic formulation of EI, LogEI has overcome this issue substantially improving the optimization performance, and reaches state-of-the-art performance compared to other recent acquisition functions [4].

3.5.2 Dealing with mixed-valued variables

Bayesian optimization as described in this chapter, requires that the parameters $x \in \chi$ of equation 3.1 are continuous. However in real life, the parameters can be purely continuous or discrete (ordinal or unordered categorical), or combination of continuous and discrete inputs. Now, the black-box function $f : \chi \times \mathbb{Z} \to \mathbb{R}$ is optimized over a compact search space $\chi \times \mathbb{Z}$, where $\mathbb{Z} = \mathbb{Z}^{(1)} \times ... \times \mathbb{Z}^{(d_z)}$ is the domain of d_z discrete parameters ($z_i \in \mathbb{Z}^{(i)}$ for $i = 1...d_z$), and $\chi = \chi^{(1)} \times ... \times \chi^{(d_x)}$ is the domain of d_x continuous parameters ($x_i \in \chi^{(i)}$ for $i = 1...d_x$).

Methods to address the mixed search space include continuous relaxation [**?**], kernel-based approaches [37] and probabilistic reparameterization [**?**]. Commonly used continuous relaxation is also applied in this thesis, and covered in this section. It means transforming categorical variables to continuous space to fit the GP surrogate model and and to optimize the acquisition function in continuous space [**?**]. As the objective function including categorical parameters cannot be evaluated with continuous values, the candidates need to be rounded back to discrete space prior to the evaluation.

In optimization of ordinal variables, variables either take integer values or are transformed to integers. For instance, temperature levels such as $\{80, 90, 100\}$, which are constrained to these integer values, serve as an example of an ordinal variable. Similarly, school grades $\{'good', 'very\ good', 'excellent'\}$ could be transformed into ordinals, for example, into $\{3, 4, 5\}$.

In optimization of categorical variables, transformation of categorical variables to continuous space is accomplished using one-hot encoded dummy variables $z'_{j,c_i} \in \{0, 1\}$ that represent presence of each category c_i of z_j:

$$z'_{j,c_i} = \begin{cases} 1 & \text{if } z_j = c_i \\ 0 & \text{otherwise} \end{cases}$$

where i is the index of the category c_i that matches with the category of parameterz_j. This encoding results in N_j dimensional sparse vector of binary variables with zero entries except for one of the dimension that equals to one. N_c corresponds to the number of categories. For a given category c_i, a categorical variable z_j with N_j categories $c_1, c_2, ..., c_{N_j}$ can be represented as follows:

$$z'_j = [0, \ldots, 0, \underbrace{1}_{\text{ith position}}, 0, \ldots, 0]$$

After encoding the discrete variables, it is possible to fit GP and compute the input values that maximize the acquisition function. However, these input values are real valued and objective function can be only evaluated at a single category at once. For ordinal variables, acquisition values can be simply rounded to the closest integer value. For categorical variables, the values of one-hot encoded categorical variables need to be transformed back to the corresponding category. The transformation is essentially a rounding that respects the limitation that only one of the categories can take value one, and rest of the entries are zero. Mathematically, let z'_j denote the one-hot encoded representation of the categorical variable z_j, and z'_{j,c_i} represents the ith element of the one-hot encoded vector. Then, the element-wise transformation $T(\cdot)$ can be defined as

$$T(z'_{j,c_i}) = \begin{cases} 1 & \text{if } i = \text{argmax}(z'_j) \\ 0 & \text{otherwise} \end{cases} \tag{3.17}$$

After the transformation, only one of the categories of z_j takes value 1 and rest are zero. The function is evaluated using the category corresponding to 1.

The problem with this naive approach is that evaluations do not actually take place at the point with the highest acquisition value. This can lead to a situation where Bayesian optimization gets stuck evaluating the same location where it has been already evaluated as it repeatedly selects infeasible continuous candidate due to its higher acquisition value, but the actual evaluation uses the transformed discrete inputs that have zero acquisition values, because they have been already evaluated [**?**].

One way to avoid the above problem is to transform real valued dummy variables back to discrete values already in a kernel function [19]:

$$k'(z'_i, z'_j) = k(T(z'_i), T(z'_j)), \tag{3.18}$$

where T denotes Transformation (3.17). Supposing a case with only a single categorical variable z_j that is encoded to one-hot dummy variables z'_j. This means that the GP surrogate have a constant value for the area that correspond to the same category c_i. For example, in case of *solvents* = (THF, Methanol, Ethanol), GP would be flat in the regions that would be transformed to the same solvent, e.g., both vectors $[0.50, 0.50, 0.99]$ and $[0.50, 0.50, 0.51]$ would be transformed to Ethanol ($[0, 0, 1]$). Thus, the distance between the vectors determined by the kernel function would be 0, which would lead to the same mean and uncertainty. Making the transformation already inside the kernel function allows the evaluation to take place at the location with the highest acquisition value. However, flat regions in response surface cause optimization of the acquisition function to be troublesome especially in larger search spaces [**?**].

3.5.3 Process-constrained Bayesian optimization

Some real-world optimizations encounter constraints regarding how parameters can be varied within the samples of the same batch. For instance, in the pipetting robot used in this study, all samples in a single batch are required to share the same temperature due to the heating plate that heats the entire rack of reaction solutions to a single temperature. This process-constraint modifies the Formula 3.1 in a way that now the parameter space χ is divided into unconstrained subspace χ_{uc} and constrained subspace χ_c , thereby expressing $\chi = \chi_{uc} \cup \chi_c$.

A straightforward method [56] for addressing the process-constraint and generating a batch $X_i = [x_{i,0}, x_{i,1}, \ldots, x_{i,K-1}]$, where i denotes the batch number and K represents the batch size, is detailed in Algorithm 2. The first member $x_{i,0}$ of the batch i is generated as usual by maximizing the chosen acquisition function. The remaining batch candidates $X_{i,k'}$, where $k' \in [1, 2, ..., K-1]$, are generated by fixing

Algorithm 2 Process-constrained batch BO

Initialize: Choose initial X_0, set iteration counter $i = 0$, define $maxIterations$ and $stoppingCriterion$

Evaluate $f(X_0)$ to obtain y_0

$D = \left\{X_0, y_0\right\}$

while $i <= maxIterations$ and $stoppingCriterion$ not met **do**

$x_{i,0} = [x^{uc}_{i,0}, x^{c}_{i,0}] = argmax_{x^c \in \chi^c} \alpha(x_{i,0}|D)$

$X^{uc}_{i,k'} = argmax_{x^{uc} \in \chi^c} \alpha(x_{i,0}|D, x^c_{i,0})$

Evaluate $f(X_i)$ to obtain y_i

$D = D \cup \left\{X_i, y_i)\right\}$

$i \leftarrow i + 1$ ▷ Move to the next iteration

end while

the constrained parameters to the value of the first candidate $x^c_{i,0}$ and then generating the unconstrained parameters of the remaining candidates $X^{uc}_{i,k'}$. The downside of this approach is that the constrained input value of the first point $x^c_{i,0}$ is optimal only for the corresponding point not for the whole batch.

3.6 Bayesian optimization in Ax and Botorch

In this study, all BO algorithms were implemented using a Python package called *Adaptive experimental platform (Ax)*. Ax provides tools for tracking experiments and wrapper functions for Bayesian optimization in BoTorch library [6]. The GP surrogate models provided by Botorch are based on *GPyTorch* [18], which uses GPU acceleration to speed up the inference.

The implemented BO algorithm used GP surrogate model with Matérn ($\nu = 5/2$) kernel, which has been found to offer promising performance in reaction optimization task [46]. Acquisition functions qEI, qLogEI and qUCB supporting parallel optimization were used. Hyperparameters are inferred from data after each batch. Noise level is considered to be unknown as experiments are typically carried out only once. Hence, also noise hyperparameter is inferred from data except for the numerical test function an arbitrary small noise level (1e-6) was used.

4. Technical solution

Automated Chemical Reaction Optimization Software (ACROS) was developed in this study. This software allows optimization of arbitrary reaction conditions and controlling a pipetting robot through a Python interface. The software is implemented in a way that it could be later integrated as part of software that more generally facilitate laboratory work [29], for example https://github.com/AaltoPML/VAI-Lab.

This chapter introduces the development principles and the main functionality by going through the software architecture.

4.1 Development

Agile software development was created to solve problems arising from lengthy development cycles of traditional software development models such as waterfall method [2]. Agile development emphasizes fast iterative development and working software over documentation.

The approach for development of ACROS can be seen as agile in many ways:

- Co-operation with a potential user: the development of software was performed in close co-operation with a chemist, who has experience in reaction optimization using OFAT and DoE approach, as well as in usage of pipetting robots.
- Incremental development: the development was divided into smaller development cycles,
- Adaptivity to changes: the solution was tested and feedback was taken into account for the development of the next version

The actual development process included 6 steps presented in Figure 4.1. First the main requirements for the solution were developed based on literature and discussions with chemists. This provided a basis for drafting a high level architecture. The first prototype was narrowed down to include only coupling reactions (a reaction where two reactants are joined). This significantly narrowed down the complexity of solution, since calculation and pipetting of reaction solution could be fixed.

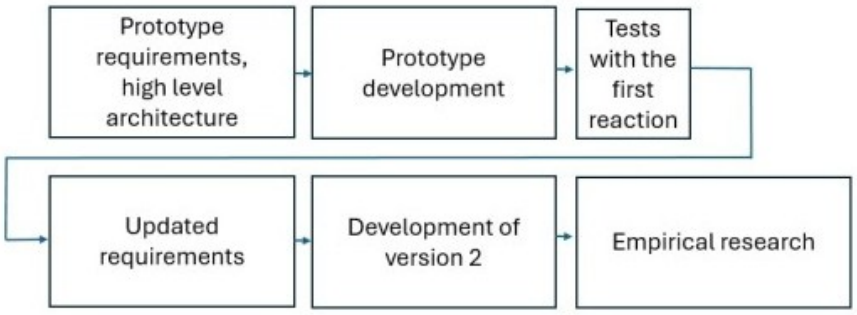

Figure 4.1: The development process of the technical solution.

The prototype was tested with optimization of Sonogashira Coupling discussed more in Section 5.3. Even though the optimization itself was not successful, the test revealed the needed improvements for developing the next version of the software. The main difference between the prototype and version 2 was that now the reaction type and number of parameters were not constrained. This software version was used in the empirical research of this thesis. Due to the time constraints of this study, further development of the software, such as improved stability and user experience, was left for the future, but the work already provided an understanding of what could be done.

4.2 Architecture

Software architecture can be viewed as a framework for satisfying software requirements [39]. This chapter first covers the high-level architecture of ACROS in Section 4.2.1 and then goes through the implemented software modules: User interface module in Section 4.2.2, Optimization module in Section 4.2.3 and OpenTrons control module in Section 4.2.4.

4.2.1 Overview

Arguably, the most important architectural decision was to separate the optimization, the control of the OpenTrons robot and the user interface from each other. This enables stand-alone usage of modules and makes program easier to maintain and develop. Using optimization module separately broadens the use cases of the system as also more complex reactions that are not supported by a commercial pipetting robot can be optimized (e.g. allowing optimization of high pressure or air sensitive reactions).

The software architecture is presented in Figure 4.2. The information flow starts from *User interface module*, which collects the information required for the initial-

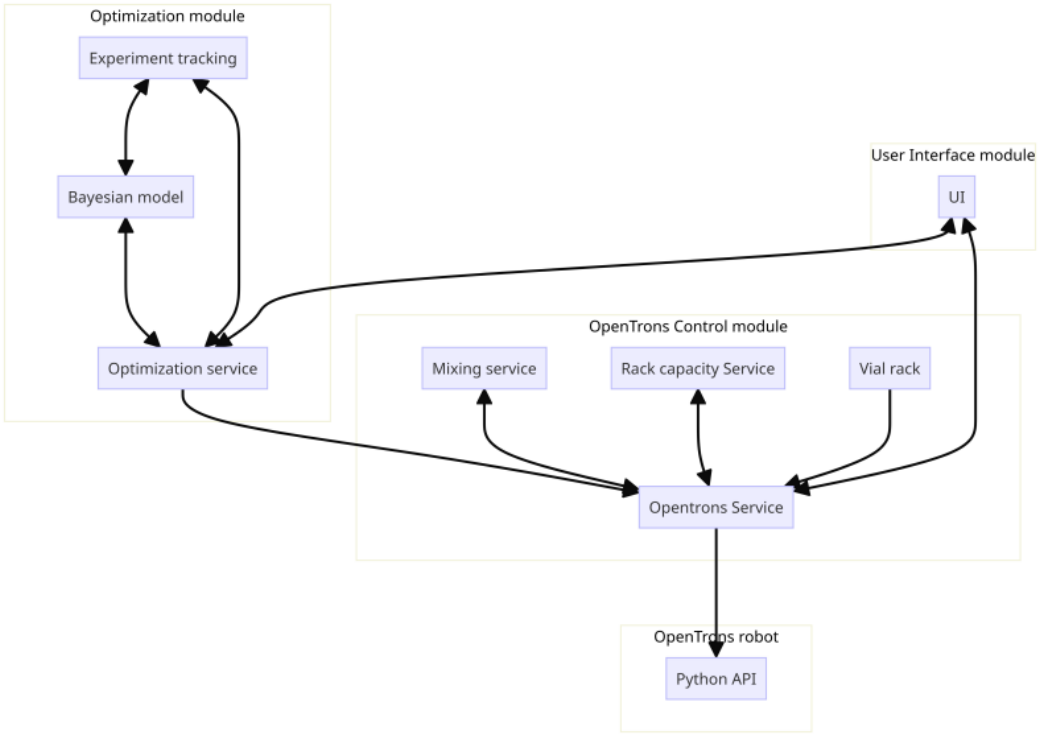

Figure 4.2: Software architecture of technical solution including main modules and their key classes

ization of the BO model. *Optimization module* initializes the model based on those specifications. For each optimization iteration that includes updating the surrogate model and generating new candidates, *Optimization module* feeds parameters of the new candidates to *OpenTrons Control module* that is responsible for interacting with *OpenTrons robot*. After each iteration, the results of the corresponding reactions are fed back to the optimization module using the user interface and underlying data sheets.

4.2.2 User interface module

The user interface as shown in Figure 4.4 allows users to enter information using Excel input files. *User interface module* has only one class *UI* representing the interface. It has different views allowing optimization using OpenTrons, standalone optimization or standalone OpenTrons usage.

To initialize the *Optimization module*, optimization settings and input file names are defined in the *Settings* view. Optimization parameters, their search spaces and chemical information such as equivalents, molar mass and density are determined in the *Parameters* view. This information is injected to the *Optimization module* to initialize the parameter space. Finally, the optimization is initialized either based on existing experimental data of the corresponding optimization setup or or without any user inputs. Next, the *UI* receives volumes of the reaction components from the

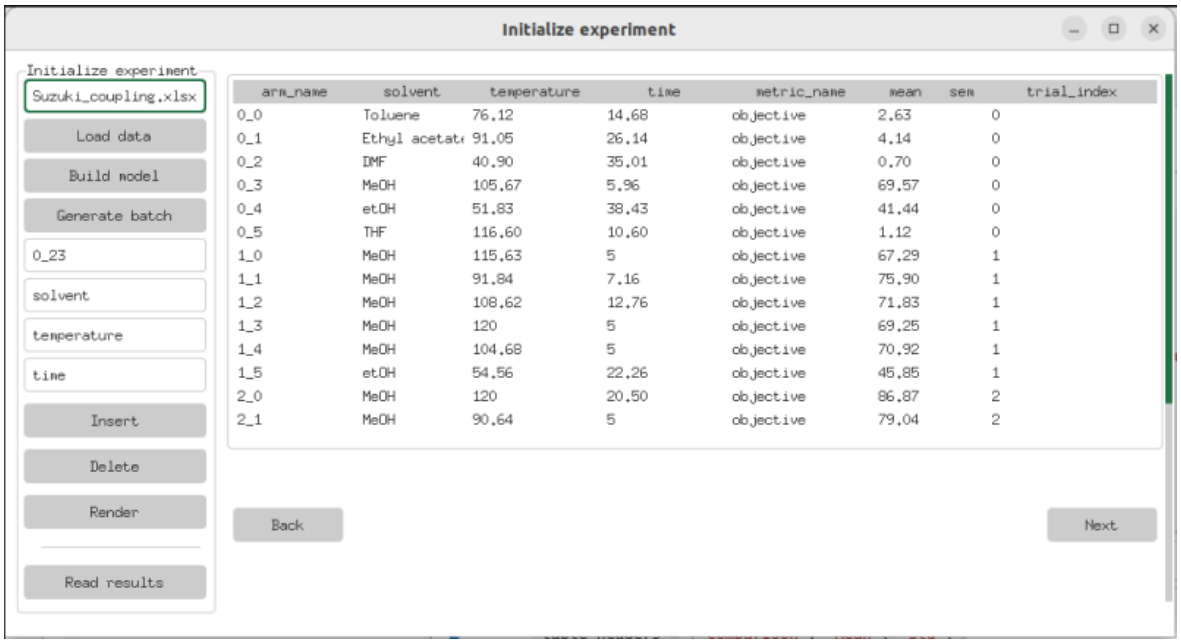

Figure 4.3: A screenshot of Experiment view of the automated optimization software. Experiment view shows the parameters and results of the experiments conducted so far and allows generating new batches and rendering the response surface

OpenTrons control module. In case of standalone OpenTrons usage, users can specify solutions on the spreadsheet and import the file via the *UI*. In the *OpenTrons* view, OpenTrons specific settings such as vial rack locations, volumes and capacities are specified and the command to generate the OpenTrons protocol is submitted.

4.2.3 Optimization module

The *Optimization module* consists of three classes illustrated in Figure 4.5: *Bayesian model*, *Experiments* and *Optimization service*. *Bayesian model* includes both a surrogate model and an acquisition function and it works as outlined in Chapter 3. The purpose of the *Experiments* class is to keep track of the experiments conducted so far and feed data to the model when needed. The role of the *Optimization service* class is to iteratively run the optimization loop described in Algorithm 1 in Section 3.1 and to interact with the other modules.

The module is designed to be used together with OpenTrons, but it can be also used as a standalone solution. When used in conjunction with OpenTrons, the parameters for the next experiments are fed to the *OpenTrons module*. In case of standalone usage, a human is responsible for carrying out reactions according to the set of reaction parameters.

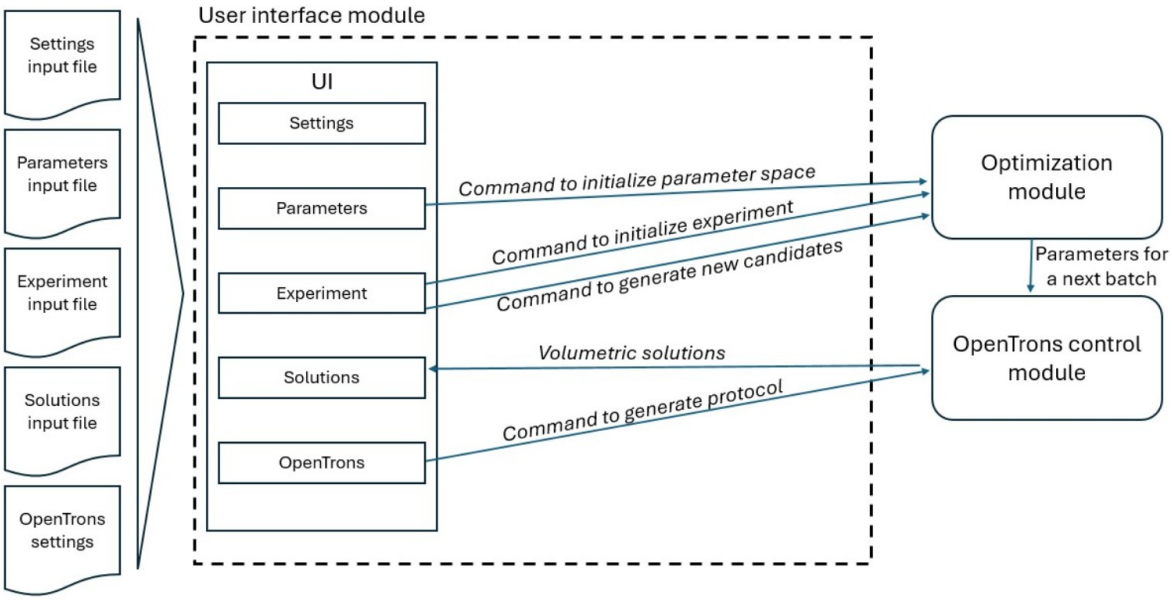

Figure 4.4: Illustration of the user interface of the technical solution

4.2.4 OpenTrons control module

OpenTrons control module governs the robot through the Python application interface (API) provided by the robot's manufacturer. The main classes of this module and their interactions are illustrated in Figure 4.6. Communication with an OpenTrons robot is facilitated using a Python protocol script.

The script includes commands to control the robot that are typically transfers of liquids from one vial to another. The transfer commands fed to the OpenTrons API are relatively simple as demonstrated in the pseudo-code:

```
transfer(from = reactant-rack(slot = 'A1'),
to = reaction-solution-rack(slot='B1'), volume=100)
```

This code would assign a pipetting operation of 100 µL from the vial in the slot 'A1' of the reactant rack to the vial in the slot 'B2' of the reaction solution rack. Because the robot cannot directly convert reaction parameters such as 1.5 mmol to volumes, a separate class *Mixing service* for handling the conversions from an optimization parameter unit (e.g.mmol) to volumetric unit (µL) is needed.

Rack capacity handler allocates chemicals to the racks holding the vials. Each vial rack has certain number of vial slots. *Rack capacity handler* allocates each chemical one or more vial(s) based on the vial size and required chemical volume. Subsequently, *OpenTrons service* uses these chemical locations for generating commands for mixing each reaction solution. The volumes of chemicals left after each mixing is also tracked by *Rack capacity handler*.

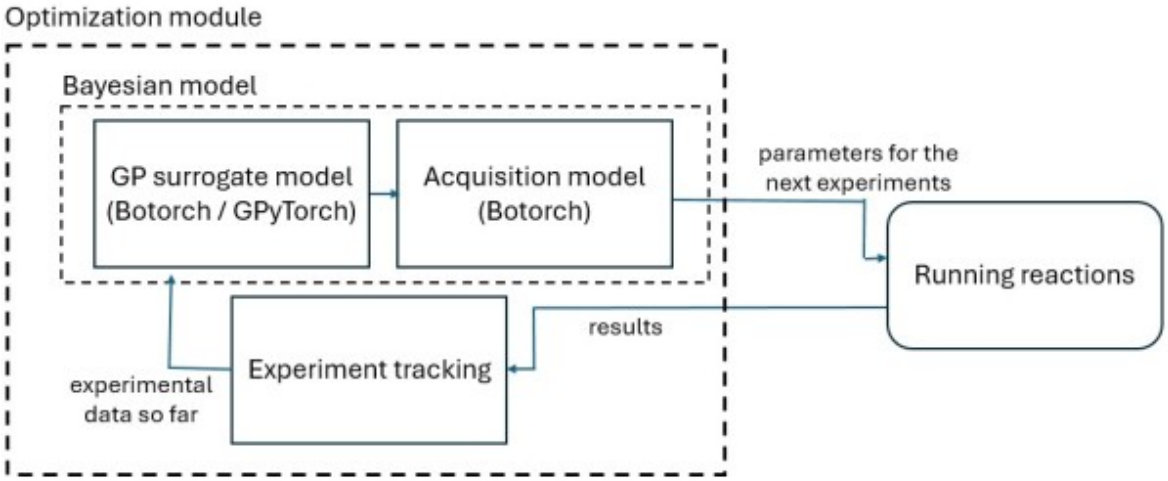

Figure 4.5: Illustration of the optimization module of the technical solution

4.3 Code and availability

The code is implemented fully in Python and available at https://github.com/Samperius/ACROS/.

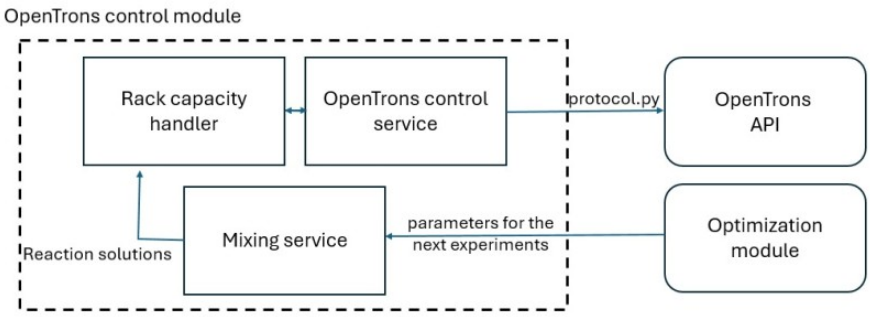

Figure 4.6: Illustration of *OpenTrons control module* of the technical solution

5. Empirical research

This chapter evaluates the performance of Bayesian optimization through empirical tests. First, the tests targeted evaluating the BO implementation of ACROS in optimization of a numerical test function. Second, the experiments aimed to study the performance of BO in reaction optimization context, particularly compared to prevalent DoE approach. The chemistry experiments were planned and conducted in co-operation with a chemist.

The numerical function experiments are covered in Section 5.1. Next, the tests using chemical reaction dataset are introduced in Section 5.2. Lastly, real-time reaction optimization experiments are presented in Section 5.3.

5.1 Numerical test function

The objective of the experiment is to validate and demonstrate the performance of implemented BO algorithm. Numerical functions are commonly used for this purpose as they can be easily evaluated, and hence, they enable conducting high number of simulations with different models. In addition, a known global optimum allows objective evaluation of the optimization outcome. Numerical test function experiments are conducted using 6-dimensional Hartmann function, which is illustrated in Figure 5.1. This function is a common choice for evaluation of BO algorithms (e.g. [4] [58] [16]).

5.1.1 Data and methods

The Hartmann 6 test is illust function is defined as follows:

$$f(\mathbf{x}) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right), \tag{5.1}$$

where c_i, a_{ij}, and p_{ij} are constants defined as:

$$c = [1, 1.2, 3, 3.2]^T$$

$$a = \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}$$

$$p = \begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.665 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix}$$

The function has 6 local minima and a single global minimum (-3.32237) at $x = (0.20169, 0.150011, 0.476874, 0.275332, 0.311652, 0.6573)$[44]. The function was optimized in the range $x_i \in [0, 1]$ for all dimensions $i = 1...6$.

Because the initialization of BO has a significant effect on the optimization results, 30 simulations with randomized initializations are run for each models. BO is tested with three different acquisition function: qUCB, qEI, and qLogEI. Batch size of five was used as it is a realistic batch size also for chemical reaction optimization where reactions are prepared manually. The same batch size is used also in real-time reaction optimization. The optimization was run for 15 batches, as for initial tests the optimization converged prior to this.

5.1.2 Results

The results in Figures 5.2 and 5.3 demonstrate the optimization performance of qUCB, qEI, and qLogEI in optimization of Hartmann 6 numerical test function . In Figure 5.2, the lines represents the cumulative minimum value of the objective value, which are labeled as the "best observed value". 30 randomly initialized experiments are shown in light grey and the mean value of the best observed values is marked in bold black line. The comparison of the acquisition functions uses mean values.

In Figure 5.3, the same optimization experiments are shown from a different perspective. Now, the lines represents the share of the optimization simulations, which cumulative minimum is below the threshold as a function of number of batches. Two thresholds were chosen, first -3.3 to demonstrate the results that are close to the global minimum, and the second threshold -3.2 to demonstrate the optimization performance in settings, where it is more important to reliably get a reasonably good results instead of reaching the global optimum.

It can be concluded that all of the acquisition functions converge close to the

global optimum. qUCB converges slightly slower, but after 10 batches it has already surpassed qEI and qLogEI. When assessing the share of the simulations reaching the threshold, the results vary depending on the threshold. More than 2/3 of simulations reach the threshold of -3.2 regardless of the chosen acquisition function. When the threshold is lowered to -3.3, half of the simulations reach the threshold. qLogEI outperforms qEI and qUCB in finding "good solution" (less than -3.2) but is worse in finding close to optimal solution(less than -3.3). In overall, the differences between the performance of the acquisition functions are minor.

5.2 Direct arylation reaction dataset

Next, the performance of the implemented BO algorithms was tested in chemical reaction optimization. The challenge in studying the performance of optimization methods using real reactions is that testing different methods would require performing the reactions separately for each different optimization configuration. To overcome this issue, a pre-collected dataset [46] of direct arylation reaction was used. Moreover, these simulations provided information on, which acquisition function to use in real-time reaction optimization in Section 5.3. It is also analyzed how BO performs compared to DoE approach.

5.2.1 Data and methods

The Arylation reaction dataset consists of 1728 reactions. Parameters are discrete consisting of: 12 ligands, four bases, four solvents, three temperature values and three concentration values. Temperatures and concentrations are treated as ordinal variables. The optimization objective is product yield (yield-%). The optimal yield of 100% is reached with a parameter combinations: ligand = CgMe-PPh, base = CsOPiv or CsOAc, Solvent = DMAc, concentration = 0.153 M, and temperature = 105 °C.

30 randomly initialized tests with three different acquisition functions are performed the same way as with the numerical test function. The batch size of 10 experiments was used to validate the performance also with higher batch size, and because that was maximum feasible batch size considering the available vial racks in OpenTrons pipetting robot. The optimization performance is tested first without any process constraints and then by constraining the temperature level for each batch as explained in Section 3.5.3.

Sometimes the BO suggests parameters that have been already evaluated, which is likely do to the rounding explained in Section 3.5.2. In these cases, the same result as in the first experiment is returned since the dataset only includes one result per

input.

BO is also compared against DoE method. Specifically, against the method explained next:

1. Experimental design is generated in two different sizes: 74 experiments and 144 experiments. General Subset Design as covered in Section 2.2.2 is used, because it allows generating design in different sizes and it has been employed effectively in optimization of categorical variables in this context [46].

2. The order of the categorical choices is shuffled since the experimental design and the evaluated inputs are dependent on the order of the categorical choices. This is because to generate GSD, the parameter space is first encoded to indexes (or levels). For example, solvents $\in \{'KOPiv','CsOAc','CsOPiv','KOAc'\}$ could be encoded to solvents $\in \{0, 1, 2, 3\}$. Hence, the generated GSD consists of experiments that are encoded to indexes that represent their corresponding categories. Figure 5.4 shows the design that is in index-encoded format and Figure 5.5 shows the same design in decoded format. This means that encoding is dependent on the order of the list of categorical choices. Since, GSD includes only a subset of all possible parameter combinations, the decoded design is dependent on the order of categorical choices. Thus, it is possible to generate random GSD designs by shuffling the order of categorical choices.

	base	ligand	solvent	concentration	temperature
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	0	2	2
3	0	0	1	0	1
4	0	0	1	1	2
...	...	...	...	...	...

Figure 5.4: Index-encoded GSD of direct arylation reaction

	0	1	2	3	4
0	KOPiv	PPh3	DMAc	0.057	90
1	CsOAc	PPhtBu2	DMAc	0.1	105
2	KOAc	BrettPhos	p-Xylene	0.1	120
3	KOPiv	PCy3 HBF4	BuCN	0.057	120
4	CsOAc	tBPh-CPhos	BuOAc	0.1	105
...	...	...	...	...	...

Figure 5.5: Decoded GSD of direct arylation reaction

3. A Quadratic model is fitted to the experimental data using Ordinary Least Squares linear regression of *Scikit learn* python library.

4. A grid-search is utilized to find the predicted maximum meaning that the model is evaluated using the parameters of 1728 reactions.

5. Finally, the reaction is evaluated with predicted optimal parameters to get the final optimization result.

5.2.2 Results of unconstrained optimization

The simulations were first run without process-constraints. This replicates stand-alone usage of the optimization module, where reactions could be prepared one-by-one allowing using of independent reaction conditions for each reactions. Figure 5.6 shows the optimization performance of the implemented BO algorithm compared to DoE approach. BO optimization results are shown in the same way as in the previous in Figure 5.2 in the previous section. The results of DoE approach use the mean value of 30 DoE simulations where the design was randomized as explained above. Figure 5.7 shows the share of simulations that are above the thresholds of 95% and 99%. In addition, the Welch t-test [59] was employed to evaluate the null hypothesis of equal mean performance compared to the results of qEI acquisition function. Table 5.1 shows descriptive and t-test statistics of the results.

BO using qEI and qLogEI reaches in average close to 100% yield in 15 batches. When computing the share of simulations that surpasses the threshold of 95% and 99% yield, qEI performs slightly better than qLogEI assuming all 15 batches are completed. qLogEI performs better when the number of batches is limited to less than 10. The mean result of qUCB is significantly worse than qEI (p-value$<$0.05). The poor performance of qUCB is likely linked to the optimization of categorical data as qUCB tended to perform well in the optimization of continuous Hartmann function. In practice, qUCB did significantly less new evaluations than other acquisition functions as it most often tended to generate already evaluated candidates. qUCB would likely to perform significantly better with an additional policy, where generating candidates that have been earlier evaluate was prevented.

DoE approach performs poorly reaching 76% mean in average. The performance of DoE slightly decreases by adding the number of experiments from 74 to 144. It appears that the response surface modeled using quadratic function is poor proxy for the true reaction function.

In overall, it can be concluded that qEI and qLogEI performance is statistically indistinguishable and they perform well in the chemical reaction optimization of cat-

Table 5.1: Descriptive and t-test statistics were performed on unconstrained optimizations within the Direct Arylation dataset, utilizing 30 randomly initialized simulations. The Welch t-test was employed to evaluate the null hypothesis of equal mean performance compared to Bayesian Optimization (BO) with the qEI acquisition function.

Comparison	Mean	Std	T-statistic	p-value
qEI	99.99	0.05	0.00	1.00
qLogEI	99.68	1.56	1.05	0.30
qUCB	98.08	4.06	2.53	0.02
qEI fixed temperature	99.91	0.27	1.48	0.15
qLogEI fixed temperature	99.30	2.37	1.55	0.13
qUCB fixed temperature	83.18	6.57	13.78	0.00
DoE 144	76.31	19.73	6.46	0.00

egorical variables. The use of qUCB in categorical settings cannot be recommended without more thorough analysis of the reasons behind the poor performance and possibly reconfiguration of the optimization algorithm. Also, the usage of DoE approach as implemented here cannot be recommended for optimization of categorical parameter based on these results.

5.2.3 Results of process-constrained optimization

The results of constrained optimization are shown in Figure 5.8 and 5.9. The figures are constructed the same way as in the unconstrained optimization. The results of constrained optimization compared to unconstrained one can be found in Table 5.1.

The results show that the mean performance of qEI and qlogEI is decreased only slightly due to the constrained temperature parameter. The difference to unconstrained optimization is statistically insignificant. However, the performance of qUCB drops significantly reaching only 83% yield in average. Now when the search space is constrained to a single temperature qUCB suggests already evaluated candidates more often. This makes the search inefficient. In overall, it can be concluded that the process-constrained optimization performs well when using with qEI and qLogEI, but the usage with qUCB would require improvements to handling the cases where the candidate has been already evaluated earlier.

5.3 Real-time chemical reaction optimizations

The target of the real-time reaction optimization experiments was to validate the optimization performance in wider range of reactions and to test the developed ACROS software in real test cases. Sonogashira coupling was optimized using purely continuous parameters, whereas Suzuki coupling was optimized using both continuous and categorical variables.

5.3.1 Data and methods

The first reaction to be optimized, Sonogashira coupling is a fundamental reaction class that are of major importance in synthetic organic chemistry [28]. The coupling of Iodobenzene and Phenylacetylene is a prototypical Sonogashira coupling [28]. More detailed explanations of the underlying chemistry are omitted as the focus of the thesis is in optimization and automation. The parameters include the concentration of Iodobenzene, catalyst loading, reaction temperature, and time. The search space of these parameters is defined as follows:

$$\begin{aligned} \text{Concentration of Iodobenzene} &\in [0.1, 0.2]\text{mmol}\,\text{mL}^{-1}, \\ \text{Catalyst loading} &\in [0.1, 0.25]\text{mg}\,\text{mL}^{-1}, \\ \text{Temperature} &\in [37, 90]^{\circ}\text{C}, \\ \text{Reaction time} &\in [60, 300]\text{min}. \end{aligned}$$

The optimization of the reaction is performed using the entire system as both Optimization and OpenTrons control module are used. The first batch is initialized using quasi-random Sobol sequence and the rest of the batches are generated using to the process-constrained BO where temperature is fixed across the samples within a batch. The first batch of reactions is completed one-by-one using the pipetting robot, because it allowed gathering information about the effects of temperature prior to starting process-constrained optimization where only one temperature level can be evaluated per batch. The catalyst is prepared separately and measured by a chemist, because the pipetting robot does not support weighting and transferring of solids. Temperature is fixed for each batch, except for the initialization, where each reaction is conducted separately. Batch size is four.

The second reaction, Suzuki coupling, is fundamental reaction in carbon to carbon bond formation utilized in wide range of applications, most prominent in drug discovery [36]. The reaction involves a coupling of an aryl boronic acid and organohalide. The parameters include the choice of solvent, reaction temperature, and time. A batch size of six is used to have each solvent tested once in the initial batch. The search space

for these parameters is defined as follows:

$$\begin{aligned} \text{Solvent} &\in \{\text{THF}, \text{DMF}, \text{MeOH}, \text{EtOAc}, \text{EtOH}, \text{Toluene}\}, \\ \text{Reaction temperature} &\in [37, 90]^{\circ}\text{C}, \\ \text{Time} &\in [5, 40]\text{min}. \end{aligned}$$

In Suzuki coupling, the reaction optimization is performed only leveraging the Optimization module as it is performed under microwave heating, since the maximum temperature of the heating plate of the pipetting robot is below the targeted reaction temperatures and the microwave heating allows significantly shorter reaction time. Now unconstrained BO was used, as the reaction solutions could be heated separately.

For both reactions, qEI was chosen as the acquisition function, because it was the best performing acquisition function when considering both the numerical test function and Direct arylation reaction optimization. Optimizations are initialized using quasi-random Sobol sequence.

Samples of the reaction solutions are analyzed using Gas Chromatography Flame Ionization Detector [9]. Using such a device in the analysis of product concentration is a standard practice in Chemistry and the procedure is described here only on high level. The device generates a chronograph that displays peaks corresponding to the separated components in the sample. The peak of intended product is identified and its area is integrated. The concentration of the product is proportional to the area of the peak. Calibration curves are used to establish the relationship between the detector response and product concentration.

5.3.2 Results

The optimization results of Sonogashira and Suzuki coupling reactions are presented in Figure 5.10. Black points represent each individual experiment. The black line denotes the cumulative maximum yield of each batch.

In optimization of Sonogashira coupling, the highest temperature tested (76.7 °C) results in the highest yield in the first batch. In the worst performing reaction, catalyst loading is low. For the next batches, BO explores higher temperatures between keeping also catalyst loading high. Concentration of iodobenzene and reaction time are varied more.

As the optimization progress to the second and third batch, the results are stuck on relatively low yield levels. In addition, close to the same reaction conditions that results in 21% yield in batch 1 results in only 8% yield in the batch 3. Thus, it is suspected that the preparation of the catalyst is causing variation to the process. As the reaction does not work as expected, the optimization is stopped after four batches.

In Suzuki coupling, the first batch of experiment provided clearly the best yield (70%) with the methanol solvent. Ethanol had the second best yield of 41%, the rest of the solvents resulting in less than 5% yields. In the next two batches, Methanol was used in 10/12 reactions. The two reactions with other solvents (Ethanol and THF) had the the lowest yields. The optimization reached the target of over 90% yield already in the third batch. As the most of the reactions were conducted using Methanol, also the reaction time and temperature were well explored with a relatively small number of experiments. Thus, it was concluded that the optimization can be stopped. The resulting response surface is shown in Figure 5.11

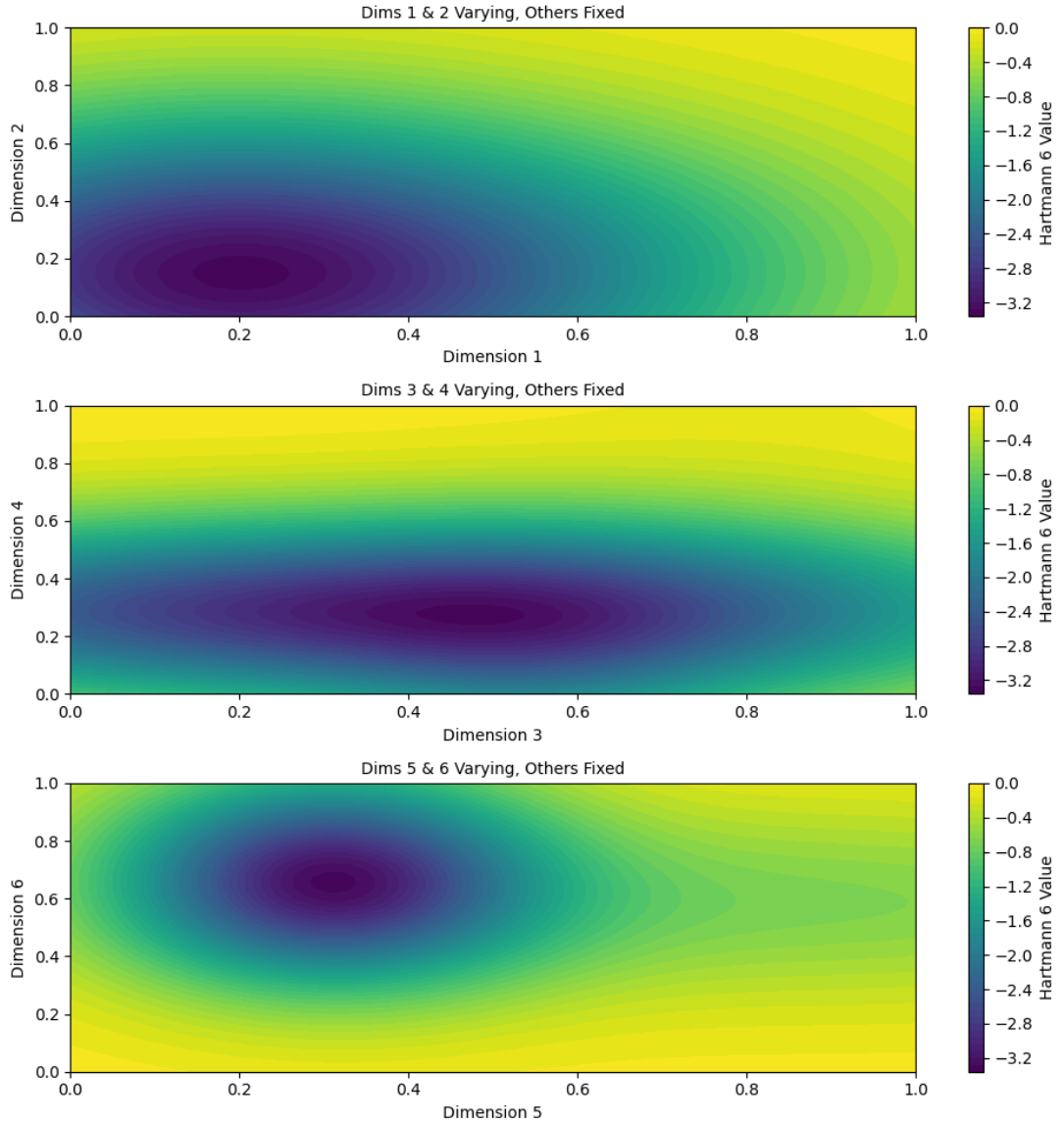

Figure 5.1: Illustration of Hartmann 6 test function. Two dimensions are varying in each figure, while others are fixed at the global minimum.

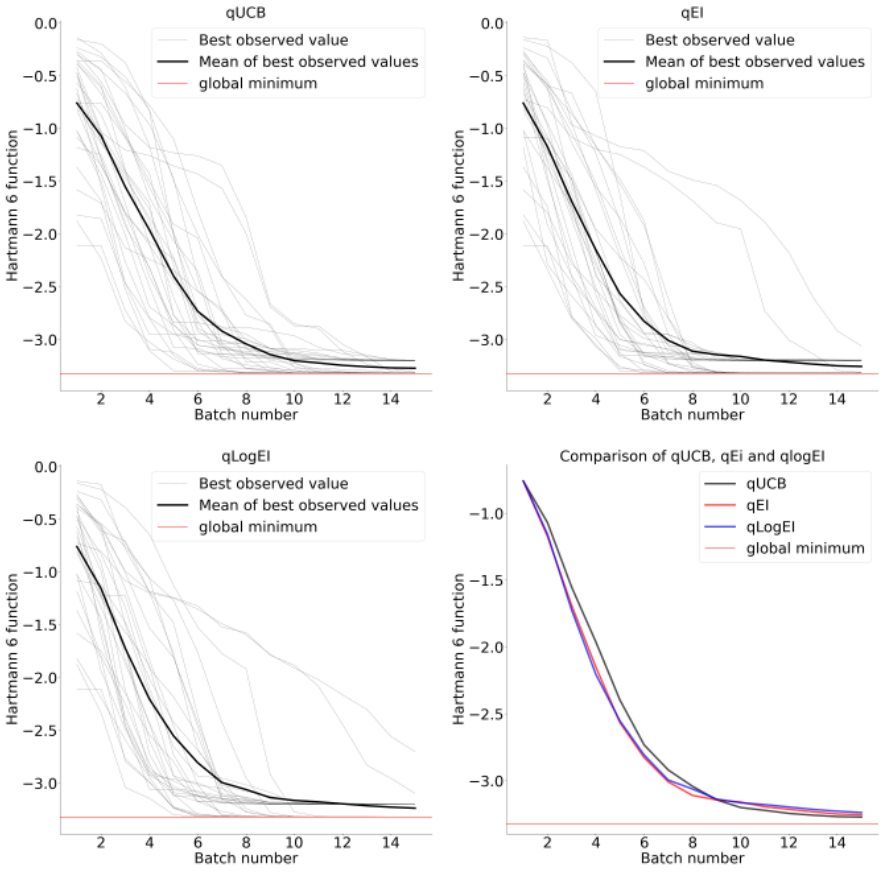

Figure 5.2: BO results of Hartmann 6 function using BO with qUCB, qEI and qlogEI and batch size of 5.

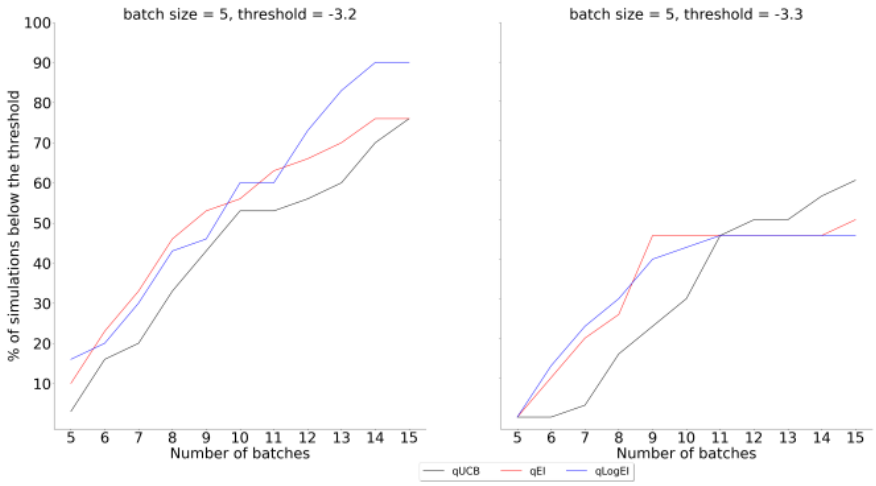

Figure 5.3: Optimization results of Hartmann 6 function using BO with qUCB, qEI and qlogEI and batch size of 5. The lines represents the share of initialized optimizations, which cumulative minimum is below the threshold as a function of number of batches.

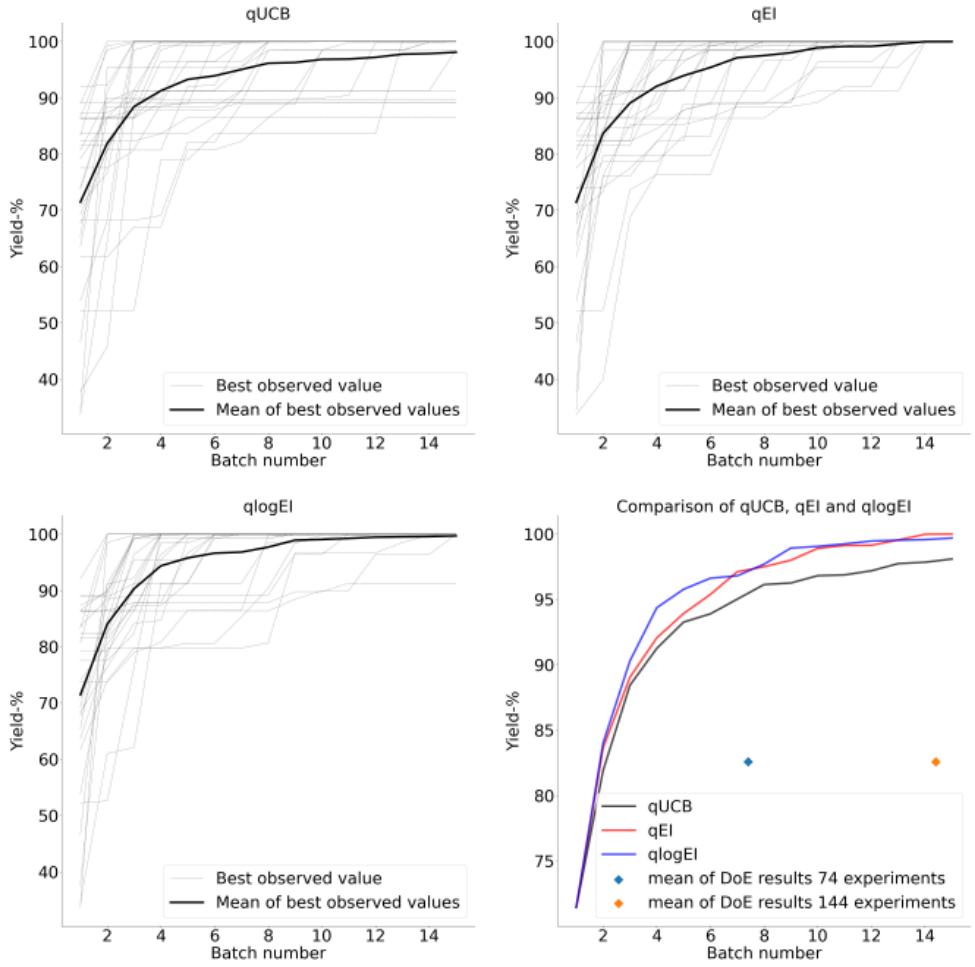

Figure 5.6: BO results of Arylation reaction optimization using BO with qUCB, qEI and qlogEI and batch size of 10. The lines represents the cumulative maximum value of the objective value. 30 randomly initialized experiments are shown in light grey and the mean is marked in bold black line. The comparison uses mean values

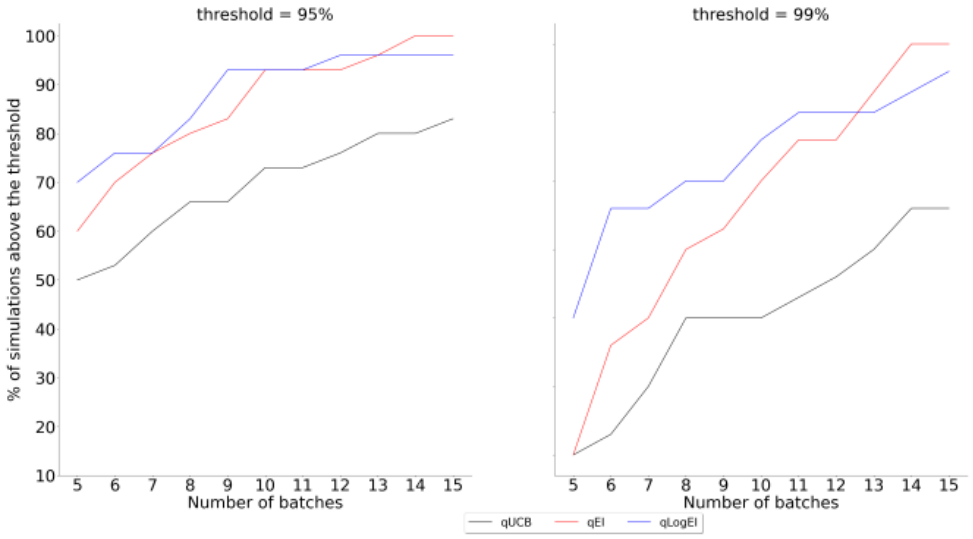

Figure 5.7: BO results of Arylation reaction optimization using BO with qUCB, qEI and qLogEI acquisition functions and a batch size of 10. The lines represents how many of 30 random initialized optimizations are below threshold.

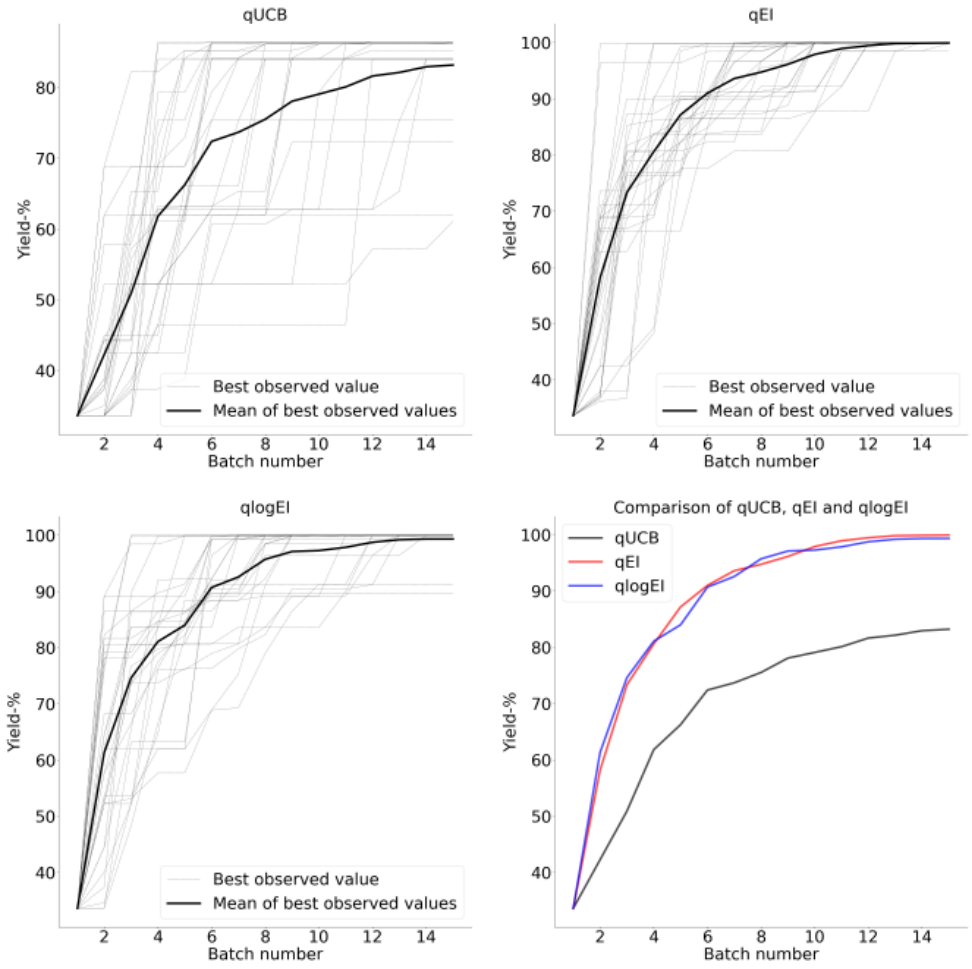

Figure 5.8: BO results of Arylation reaction optimization using process-constrained BO with qUCB, qEI and qlogEI and batch size of 10.

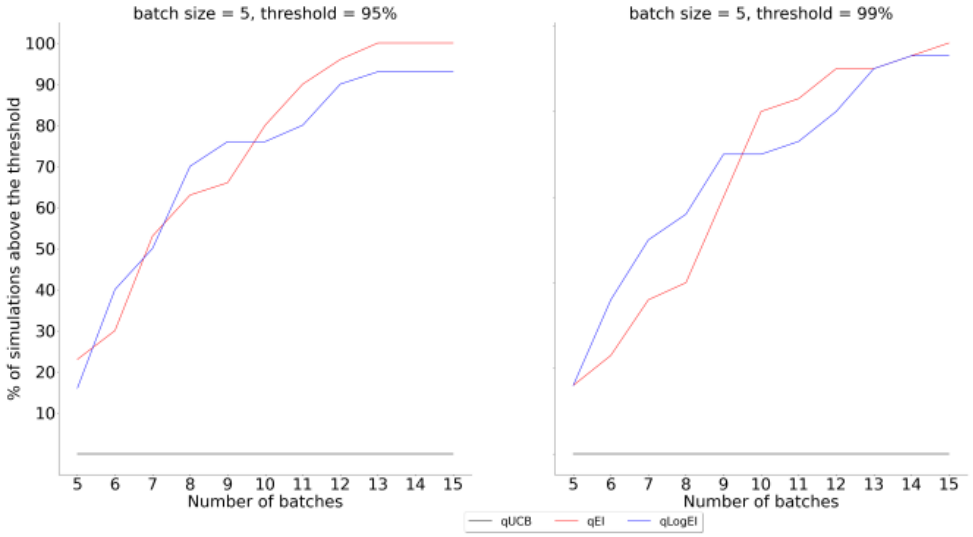

Figure 5.9: BO results of Arylation reaction optimization using BO with qUCB, qEI and qlogEI and batch size of 5.

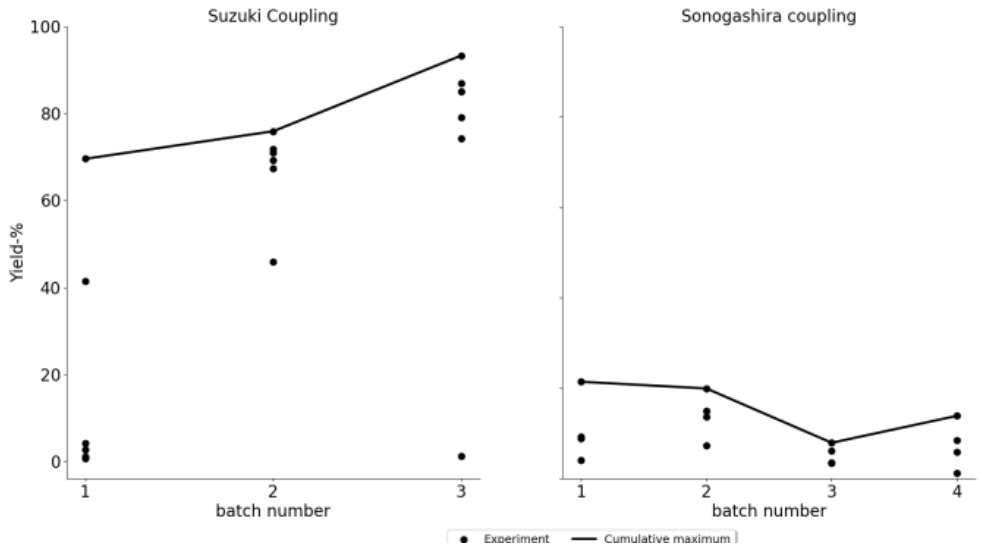

Figure 5.10: BO results of Suzuki and Sonogashira reaction optimizations.

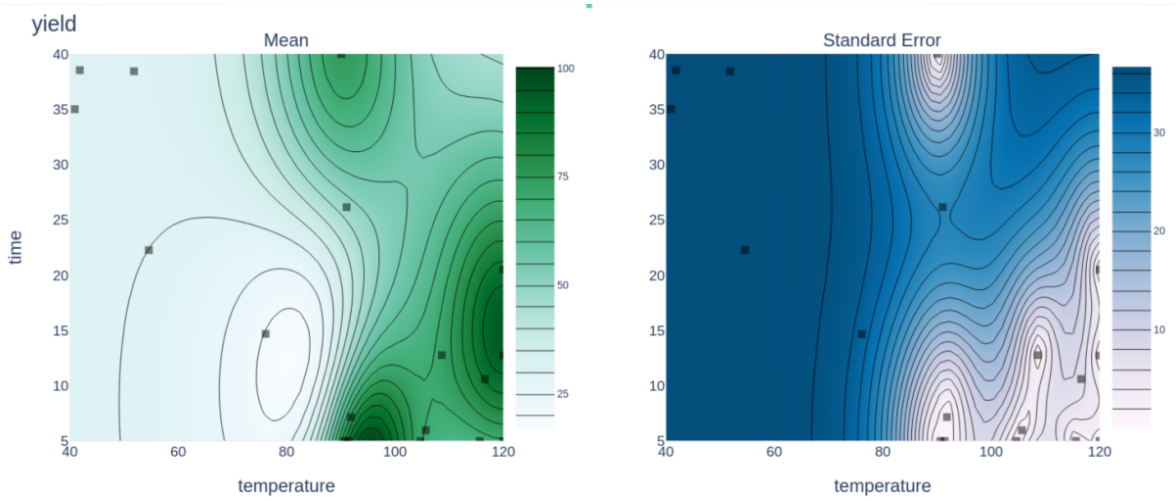

Figure 5.11: A screenshot of Suzuki response surface plotted using ACROS.

6. Conclusions

Bayesian Optimization (BO) is a popular method for optimizing objective functions that are costly and time-consuming to evaluate. BO is a sequential algorithm, where the objective function is approximated using a surrogate model. The next parameters to evaluate are generated by maximizing an acquisition function, that guides the search towards the most promising areas of the search space [27]. It is commonly used for tasks like hyperparameter tuning in machine learning, engineering systems design, and material and drug discovery [17]. In this study, BO is applied for optimization of chemical reactions. This typically means finding reaction conditions, such as reaction temperature, time and concentrations, that produce the highest yield.

The use of automation is becoming more popular in chemistry. Tools such as flow reactors and pipetting robots are examples of the rise of automation in chemistry [12]. Automation approaches such as high-throughput experimentation target increasing the number of experiments conducted per day. However, required time for reaction optimization is determined by both the time consumed per experiment and the number of experiments required. In this study, the attempt is to minimize both of these factors. Hence, the target of this study is to combine pipetting robot with BO.

This study presents, *ACROS*, a system that integrates Bayesian optimization with automation. This automated optimization system establishes an iterative loop between a pipetting robot and a BO algorithm, having potential to substantially reduce a chemist's workload and overall time spent on reaction optimization. Furthermore, this study demonstrated that BO can be implemented as a user friendly software that does not require substantial programming or statistical knowledge. This is crucial as one of the key factors behind the popularity of DoE approach has been the availability of easy-to-use optimization software [24].

The efficacy of Bayesian optimization in chemical reaction optimization was investigated. The study demonstrated that BO can efficiently identify optimal or near-optimal reaction conditions compared to traditional Design of Experiment methods. The performance of three acquisition functions; qUCB, qEI, and qlogEI; is compared. The findings reveal that qEI and qLogEI reach better results in optimization of categorical variables in Arylation experiment. The optimization of Suzuki coupling demon-

strates that optimization software can efficiently facilitate reaction optimization.

For user friendliness and better stability of the program, it should be tested with more diverse set of reactions and with new users. Software could also support wider variety of measurement units. To maintain the software, an automated testing would be beneficial to detect if the robot's updates have effect on the software. In addition, the developed system still requires human input in the optimization loop and has constraints on what types of reactions (e.g. open air, only liquids). This can limit the practical benefits of using the system as whole. To overcome at least some of these limitations, more advanced robot could be used. Features such as automated capping of the vials and solid dispensing would broaden the range of possible reactions. Smaller pipetting volumes would reduce the need of stock solutions and allow automatic pipetting of analysis samples, which typically need to have very low concentration. Finally, an integrated analysis equipment would obviate the need of a human analyzing the samples in separate analysis device and feeding the results back to the system.

In future, Bayesian optimization has potential to become a standard solution for reaction optimization. One of its main benefits comes from its flexibility. The starting parameters can be chosen based on chemist's intuition, and thus, getting a quicker start to the optimization than in random initialization. Another aspect of the flexibility is a possibility to stop the optimization at any point and still to be able to interpret the response surface and its uncertainty. Currently used DoE approach and OFAT method require user to complete the entire optimization campaign as planned to fully interpret the results.

Another advantage of BO is its extensibility. To further develop the optimization software, BO could be extended to the following areas:

- Multi-objective Bayesian optimization consider optimization of multiple objectives, which are conflicting in a way that they all of them cannot be optimized simultaneously [7]. For example, purely optimizing product yield may lead to a situation where some of the reagents convert to undesired products meaning the loss of starting materials and impurity in the reaction solution. Therefore, incorporating multiple objectives can provide a more comprehensive optimization approach.

- Multi-fidelity BO considers using secondary information sources, that are typically cheaper to evaluate but less accurate, to enhance the efficiency of BO [49]. Reaction databases are possible information sources that could add value to the optimization.

- Human-in-the-loop machine learning is a technique, where humans and machine learning algorithms interact [34]. Integrating human into the BO loop could

mean using tacit knowledge of a chemist to improve optimization performance. For example, in material optimization human-in-the-loop approach has been used to improve the trustworthy of BO by integrating human into the loop to comment the quality of samples.

Overall, it can be concluded that chemists should consider using Bayesian Optimization (BO) in chemical reaction optimization due to its performance, robustness, and flexibility. However, broader adoption of BO will likely require the further development of user-friendly optimization tools and increased education on its application and benefits.

Bibliography

[1] For chemists, the AI revolution has yet to happen. *Nature*, 617:438–438, 2023.

[2] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta. Agile software development methods: Review and analysis. *arXiv preprint arXiv:1709.08439*, 2017.

[3] M. Altamirano, F.-X. Briol, and J. Knoblauch. Robust and conjugate Gaussian process regression. *arXiv preprint arXiv:2311.00463*, 2023.

[4] S. Ament, S. Daulton, D. Eriksson, M. Balandat, and E. Bakshy. Unexpected improvements to expected improvement for Bayesian optimization. *arXiv preprint arXiv:2310.20708*, 2023.

[5] M. Balandat, B. Karrer, D. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy. Botorch: A framework for efficient monte-carlo Bayesian optimization. *Advances in neural information processing systems*, 33:21524–21538, 2020.

[6] M. Balandat, B. Karrer, D. R. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*, 2020.

[7] S. Belakaria, A. Deshwal, and J. R. Doppa. Max-value entropy search for multi-objective Bayesian optimization. *Advances in neural information processing systems*, 32, 2019.

[8] M. A. Bezerra, R. E. Santelli, E. P. Oliveira, L. S. Villar, and L. A. Escaleira. Response surface methodology (RSM) as a tool for optimization in analytical chemistry. *Talanta*, 76(5):965–977, 2008.

[9] A. T. Blades. The flame ionization detector. *Journal of Chromatographic Science*, 11(5):251–255, 1973.

[10] P. Bratley and B. L. Fox. Algorithm 659: Implementing sobol's quasirandom sequence generator. *ACM Transactions on Mathematical Software (TOMS)*, 14(1):88–100, 1988.

[11] M. Christensen, L. P. Yunker, F. Adedeji, F. Häse, L. M. Roch, T. Gensch, G. dos Passos Gomes, T. Zepel, M. S. Sigman, A. Aspuru-Guzik, et al. Data-science driven autonomous process optimization. *Communications Chemistry*, 4(1):112, 2021.

[12] M. Christensen, L. P. Yunker, P. Shiri, T. Zepel, P. L. Prieto, S. Grunert, F. Bork, and J. E. Hein. Automation isn't automatic. *Chemical science*, 12(47):15473–15490, 2021.

[13] B. Durakovic. Design of experiments application, concepts, examples: State of the art. *Periodicals of Engineering and Natural Sciences*, 5(3), 2017.

[14] M. Feurer, B. Letham, and E. Bakshy. Scalable meta-learning for Bayesian optimization. 2018.

[15] M. Feurer, J. Springenberg, and F. Hutter. Initializing bayesian hyperparameter optimization via meta-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[16] P. Frazier, W. Powell, and S. Dayanik. The knowledge-gradient policy for correlated normal beliefs. *INFORMS journal on Computing*, 21(4):599–613, 2009.

[17] P. I. Frazier. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[18] J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson. Gpytorch: Blackbox matrix-matrix Gaussian process inference with gpu acceleration. *Advances in neural information processing systems*, 31, 2018.

[19] E. C. Garrido-Merchán and D. Hernández-Lobato. Dealing with categorical and integer-valued variables in Bayesian optimization with Gaussian processes. *Neurocomputing*, 380:20–35, 2020.

[20] A. Graves. Practical variational inference for neural networks. *Advances in neural information processing systems*, 24, 2011.

[21] C. He, C. Zhang, T. Bian, K. Jiao, W. Su, K.-J. Wu, and A. Su. A review on artificial intelligence enabled design, synthesis, and process optimization of chemical products for industry 4.0. *Processes*, 11(2):330, 2023.

[22] A. Hebbal, M. Balesdent, L. Brevault, N. Melab, and E.-G. Talbi. Deep Gaussian process for multi-objective Bayesian optimization. *Optimization and Engineering*, 24(3):1809–1848, 2023.

[23] E. Heid, K. P. Greenman, Y. Chung, S.-C. Li, D. E. Graff, F. H. Vermeire, H. Wu, W. H. Green, and C. J. McGill. Chemprop: A machine learning package for chemical property prediction. *Journal of Chemical Information and Modeling*, 64(1):9–17, 2023.

[24] D. B. Hibbert. Experimental design in chromatography: a tutorial review. *Journal of chromatography B*, 910:2–13, 2012.

[25] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization*, pages 507–523. Springer, 2011.

[26] W. A. Jensen. Confirmation runs in design of experiments. *Journal of Quality Technology*, 48(2):162–177, 2016.

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

[28] V. K. Kanuru, G. Kyriakou, S. K. Beaumont, A. C. Papageorgiou, D. J. Watson, and R. M. Lambert. Sonogashira coupling on an extended gold surface in vacuo: reaction of phenylacetylene with iodobenzene on au (111). *Journal of the American Chemical Society*, 132(23):8081–8086, 2010.

[29] A. Klami, T. Damoulas, O. Engkvist, P. Rinke, and S. Kaski. Virtual laboratories: Transforming research with ai. *Data-Centric Engineering*, 2024.

[30] S. Kucherenko, D. Albrecht, and A. Saltelli. Exploring multi-dimensional spaces: A comparison of latin hypercube and quasi monte carlo sampling techniques. *arXiv preprint arXiv:1505.02350*, 2015.

[31] H. J. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 1964.

[32] J. L. Love. Chemical metrology, chemistry and the uncertainty of chemical measurements. *Accreditation and quality assurance*, 7:95–100, 2002.

[33] M. McKay, R. Beckman, and W. Conover. Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[34] E. Mosqueira-Rey, E. Hernández-Pereira, D. Alonso-Ríos, J. Bobes-Bascarán, and Á. Fernández-Leal. Human-in-the-loop machine learning: a state of the art. *Artificial Intelligence Review*, 56(4):3005–3054, 2023.

[35] K. P. Murphy. *Probabilistic machine learning: Advanced topics*. MIT press, 2023.

[36] M. R. Netherton, C. Dai, K. Neuschütz, and G. C. Fu. Room-temperature alkyl-alkyl suzuki cross-coupling of alkyl bromides that possess β hydrogens. *Journal of the American Chemical Society*, 123(41):10099–10100, 2001.

[37] C. Oh, J. Tomczak, E. Gavves, and M. Welling. Combinatorial Bayesian optimization using the graph cartesian product. *Advances in Neural Information Processing Systems*, 32, 2019.

[38] A. J. Parrott, R. A. Bourne, G. R. Akien, D. J. Irvine, and M. Poliakoff. Self-optimizing continuous reactions in supercritical carbon dioxide. *Angewandte Chemie International Edition*, 50(16):3788–3792, 2011.

[39] D. E. Perry and A. L. Wolf. Foundations for the study of software architecture. *ACM SIGSOFT Software engineering notes*, 17(4):40–52, 1992.

[40] C. Pirola, I. Rossetti, V. Ragaini, et al. Are conversion, selectivity and yield terms unambiguously defined in chemical and chemical engineering terminology? *Sci. Technol*, 2006.

[41] M. Poroch-Seritan, S. Gutt, G. Gutt, I. Cretescu, C. Cojocaru, and T. Severin. Design of experiments for statistical modeling and multi-response optimization of nickel electroplating process. *Chemical Engineering Research and Design*, 89(2):136–147, 2011.

[42] J. Ren and D. Sweet. Optimal initialization of batch Bayesian optimization. *arXiv preprint arXiv:2404.17997*, 2024.

[43] M. Renardy, L. R. Joslyn, J. A. Millar, and D. E. Kirschner. To sobol or not to sobol? the effects of sampling schemes in systems biology applications. *Mathematical biosciences*, 337:108593, 2021.

[44] B. Shahriari, A. Bouchard-Côté, and N. Freitas. Unbounded Bayesian optimization via regularization. In *Artificial intelligence and statistics*, pages 1168–1176. PMLR, 2016.

[45] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[46] B. J. Shields, J. Stevens, J. Li, M. Parasram, F. Damani, J. I. M. Alvarado, J. M. Janey, R. P. Adams, and A. G. Doyle. Bayesian reaction optimization as a tool for chemical synthesis. *Nature*, 590(7844):89–96, 2021.

[47] J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

[48] I. M. Sobol. On the distribution of points in a cube and the approximate evaluation of integrals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 7(4):784–802, 1967.

[49] J. Song, Y. Chen, and Y. Yue. A general framework for multi-fidelity Bayesian optimization with Gaussian processes. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3158–3167. PMLR, 2019.

[50] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

[51] N. T. Stevens and C. M. Anderson-Cook. Design and analysis of confirmation experiments. *Journal of Quality Technology*, 51(2):109–124, 2019.

[52] I. Surowiec, L. Vikstrom, G. Hector, E. Johansson, C. Vikstrom, and J. Trygg. Generalized subset designs in analytical chemistry. *Analytical chemistry*, 89(12):6491–6497, 2017.

[53] C. J. Taylor, A. Pomberger, K. C. Felton, R. Grainger, M. Barecka, T. W. Chamberlain, R. A. Bourne, C. N. Johnson, and A. A. Lapkin. A brief introduction to chemical reaction optimization. *Chemical Reviews*, 123(6):3089–3126, 2023.

[54] J. K. Telford. A brief introduction to design of experiments. *Johns Hopkins apl technical digest*, 27(3):224–232, 2007.

[55] W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.

[56] P. Vellanki, S. Rana, S. Gupta, D. Rubin, A. Sutti, T. Dorin, M. Height, P. Sanders, and S. Venkatesh. Process-constrained batch Bayesian optimisation. *Advances in Neural Information Processing Systems*, 30, 2017.

[57] A. I. Vogel. Practical organic chemistry. *Long Man Group Lted, London*, 1974.

[58] J. Wang, S. C. Clark, E. Liu, and P. I. Frazier. Parallel Bayesian global optimization of expensive functions. *Operations Research*, 68(6):1850–1865, 2020.

[59] B. L. Welch. The significance of the difference between two means when the population variances are unequal. *Biometrika*, 29(3/4):350–362, 1938.

[60] C. K. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[61] Z. Zhou, X. Li, and R. N. Zare. Optimizing chemical reactions with deep reinforcement learning. *ACS central science*, 3(12):1337–1344, 2017.