Trading Off Quality and Uncertainty Through Multi-Objective Optimisation in Batch Bayesian Optimisation

Chao Jiang and Miqing Li*

School of Computer Science, University of Birmingham, Birmingham, United Kingdom
cxj249@student.bham.ac.uk, m.li.8@bham.ac.uk

Abstract

Batch Bayesian Optimisation (BBO) has emerged as a potent approach for optimising expensive black-box functions. Central to BBO is the issue of selecting a number of solutions at the same time through a batch method, in the hope for them to represent good, yet different, trade-offs between exploitation and exploration. To address this issue, one of the recent advancements has leveraged multi-objective optimisation to simultaneously consider several acquisition functions (e.g., PI, EI, and LCB), allowing them to complement each other. However, acquisition functions may behave similarly (since they all aim for a good balance between exploitation and exploration), restricting the search on different promising areas. In this paper, we attempt to address the above issue. We directly treat exploitation (reflected by quality, i.e., the posterior mean) and exploration (reflected by uncertainty) as two objectives. When selecting trade-off solutions between the two objectives, we consider a dynamically updated Pareto front where the uncertainty changes once a solution is selected, thereby allowing exploration on different promising areas. Through an extensive experiment study, we show the effectiveness of the proposed method in comparison with state-of-the-arts in the area.

Introduction

Bayesian Optimisation (BO) is a successful technique for solving expensive black-box problems (Wang et al. 2017a; Eriksson et al. 2019; Chugh 2020; Song et al. 2022; Huang et al. 2024; Santoni et al. 2024). In BO, the search needs to achieve a trade-off between exploitation and exploration during the optimisation process. The former involves refining and leveraging high-quality solutions that have already been identified, while the latter entails exploring less-known areas. In BO, an acquisition function plays a major role in achieving the trade-off between exploitation and exploration. Notable examples of such acquisition functions include Probability of Improvement (PI) (Kushner 1964), Expected Improvement (EI) (Mockus, Tiesis, and Zilinskas 1978), and Lower Confidence Bound (LCB) (Lai and Robbins 1985).

In practical scenarios, it is common to leverage hardware capabilities to conduct a number of evaluations in parallel, which is particularly beneficial when dealing with expensive problems. As a result, batch (parallel) Bayesian optimisation (BBO) has been developed (Schonlau 1997), in which q solutions are selected for evaluation in parallel by using a batch method. By evaluating q solutions in parallel, BBO can significantly reduce the overall optimisation time. However, in BBO the task of balancing exploitation and exploration is non-trivial as one may not want to find a batch of adjacent solutions that represent similar trade-offs between exploitation and exploration.

In response to the challenge faced in batch methods, one of the recent advancements started making use of multiple acquisition functions (Bischl et al. 2014; Feng et al. 2015; Grobler, Kok, and Wilke 2017; Lyu et al. 2018; Chen, Luo, and Wang 2022; Chen et al. 2023). These studies leverage multi-objective optimisation to simultaneously consider several acquisition functions, thereby allowing a complement between them. One benefit of considering multiple acquisition functions in BBO is that a number of solutions having different preferences with respect to exploitation and exploration can be selected in a batch.

However, despite having their own preferences, acquisition functions may behave similarly since they all aim to achieve a good balance between exploitation and exploration. This may lead to the selection of similar solutions for evaluation. Figure 1 illustrates this situation, where acquisition functions PI, EI, and LCB are considered. As can be seen from the figure, the Pareto fronts of all the combinations of these acquisition functions are located within a narrow area (marked by a red line). Keeping searching around this area may not help in exploring undeveloped, yet potentially promising, areas.

In this paper, we attempt to address the above issue by directly treating exploitation and exploration as two objectives, reflected by quality (i.e., the posterior mean) and uncertainty, respectively. These two very different objectives enable the search to have chances to explore diverse yet potentially promising areas. As can be seen from Figure 1, the Pareto front of exploitation (i.e., $\mu(x)$) and exploration ($\sigma(x)$) consists of points on multiple areas (cyan points in the top panel), including some far from the Pareto front area of the three acquisition functions (red line). We call the proposed method POEE (Pareto optimisation for exploitation and exploration), and its main contributions are as follows.

*Corresponding author.

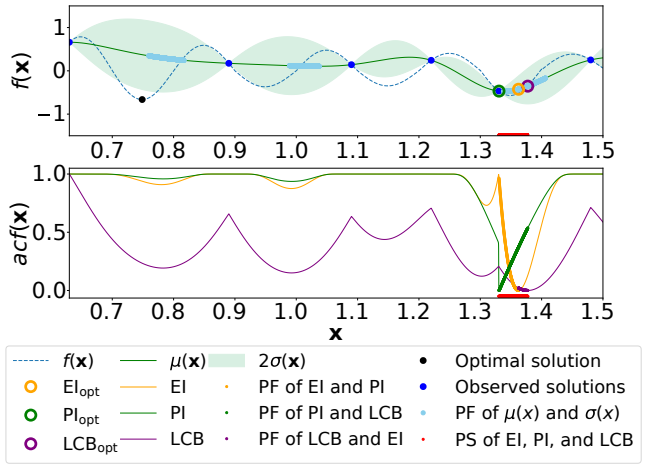


Figure 1: An example of the Pareto fronts (PFs) of different objectives (e.g., PI, EI, and LCB). For simplicity, all objectives are considered to be minimised. Top: The blue dashed line shows the black-box function approximated by the Gaussian Process. The dark green line shows the mean $\mu(x)$ and the light green area represents twice the posterior standard deviation $\sigma(x)$. Yellow, green, and purple circles are optimal solutions of EI, PI, and LCB, respectively. Blue points are observed solutions and cyan points correspond to the Pareto front of $\mu(x)$ and $\sigma(x)$. Bottom: Landscapes of the three normalised acquisition functions, EI (yellow line), PI (green line), and LCB (purple line), are shown. The Pareto fronts, derived from combinations of these acquisition functions, are denoted by yellow (for EI and PI), green (for PI and LCB), and purple (for LCB and EI) solutions. Notably, the Pareto fronts of all combinations of the acquisition functions are located within a narrow area in the right of the figure, i.e., the red area, while the problem's real optimal solution (black point) sits at the very left of the figure.

- POEE considers Pareto optimisation with respect to quality (i.e., the posterior mean) and uncertainty. This is unlike most Pareto optimisation-based batch methods (Bischl et al. 2014; Feng et al. 2015; Grobler, Kok, and Wilke 2017; Lyu et al. 2018; Chen, Luo, and Wang 2022; Chen et al. 2023) that simultaneously consider multiple acquisition functions, e.g., PI, EI, and LCB. This avoids selecting similar solutions in a batch.
- POEE considers a dynamically updated quality-uncertainty Pareto front, which changes after each solution is selected (since the uncertainty changes). This is different from existing studies that either only consider one solution from the quality-uncertainty Pareto front (De Ath et al. 2020; De Ath, Everson, and Fieldsend 2021), or select a batch of solutions from the quality-uncertainty Pareto front in a one-off way (Gupta et al. 2018; Binois, Collier, and Ozik 2021). The constantly changing Pareto front allows the exploration on different trade-off areas.

We evaluate our method by comparing it with twelve well-established methods on 14 synthetic and practical problems. Further experimental studies, such as ablation and parameter sensitivity, have been carried out to help understand the proposed method. The code, data, and supplementary material are available at https://github.com/ChaoJiang52/AAAI-POEE.

Preliminaries

Bayesian Optimisation

We want to solve an expensive black-box optimisation problem: $x^* = \arg\min_{x \in \mathcal{X}} f(x)$, where $x \in \mathcal{X}$ denotes variables in decision space; f is a black-box function; $\mathcal{X} \subseteq R^d$ denotes a compact set.

BO algorithm is a randomised algorithm tailored for addressing expensive black-box optimisation problems (Garnett 2023), in which solutions are sequentially selected. In each iteration, a Gaussian process model is trained using the data set D. An acquisition function $acf(\mathbf{x})$ is then optimised to identify the next solution x_t for evaluation. Upon selecting x_t, it is evaluated, and the data set D is augmented.

Gaussian processes Gaussian process regression, a prevalent method for constructing a meta-model, is based on the Gaussian process (MacKay 1998). Given N training solutions $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ where $y_i = f(x_i) + \epsilon_i$ with $\epsilon_i \sim \mathcal{N}(0, \sigma_\epsilon^2)$, the posterior distribution[1] at a new location x is a Gaussian distribution:

$$p(y|\mathcal{D}, \theta, x) = \mathcal{N}(\mu(x), \sigma^2(x)) \quad (1)$$

$$\mu(x) = \mathbf{k}(x, X)(\mathbf{K} + \sigma_\epsilon^2 \mathbf{I})^{-1}\mathbf{y} \quad (2)$$

$$\sigma^2(x) = k(x, x) - \mathbf{k}(x, X)^T(\mathbf{K} + \sigma_\epsilon^2 \mathbf{I})^{-1}\mathbf{k}(X, x) \quad (3)$$

where $X \in R^{N \times d}$ and $\mathbf{y} \in R^N$ denote the matrix of evaluated solutions and the corresponding vector of function values, respectively; θ represents the kernel hyperparameters; $k(x, X)$ denotes the covariance vector between x and each element in X, $K \in R^{N \times N}$ is the covariance matrix, and $k(x, x)$ denotes the covariance between x and itself.

Acquisition functions After training a Gaussian process regression model, an acquisition function is optimised to select a suitable solution for evaluation (Wang et al. 2023). The most commonly used acquisition functions are LCB and EI.

LCB (Lai and Robbins 1985) evaluates a solution based on its predicted mean and the associated uncertainty: $acf_{LCB}(x) = \mu(x) - \sqrt{\beta_t}\sigma(x)$, where $\sqrt{\beta_t} \geq 0$ represents the weight, typically contingent upon the current iteration t.

EI (Mockus, Tiesis, and Zilinskas 1978; Jones, Schonlau, and Welch 1998) is another prevalent acquisition function, defined as $EI(x) = \sigma(x)(\lambda\Phi(\lambda) + \phi(\lambda))$, where $\lambda = \frac{y^* - \mu(x)}{\sigma(x)}$; y^* is the best observed y value. Φ and ϕ denote the standard normal cumulative distribution function and probability density function, respectively.

[1]Note that the matrix inversion required by Equations (2) and (3) uses Cholesky decomposition (Williams and Rasmussen 2006).

Batch Bayesian Optimisation

In practical applications, it is typical to utilise hardware capabilities to perform multiple evaluations concurrently, which is advantageous for expensive problems. Consequently, BBO has been introduced (Schonlau 1997), where q solutions are chosen through a batch method for simultaneous evaluation. The primary aim of the BBO algorithm is to balance exploitation and exploration by selecting q trade-off solutions for concurrent evaluation, thereby reducing the overall optimisation time.

Several acquisition functions have been proposed to determine q promising batch solutions. Ginsbourger, Le Riche, and Carraro (2008); Chevalier and Ginsbourger (2013); Shah and Ghahramani (2015); Wu and Frazier (2016) jointly estimate the batch of solutions, which may scale up poorly with the batch size (Daxberger and Low 2017). Thus, iteratively picking q batch solutions has emerged as the prevalent methodology. The stochastic acquisition function (Hunt 2020), such as Thompson Sampling (Kandasamy et al. 2018), is naturally capable of producing several unique solutions. Information-based methods (Wang and Jegelka 2017; Moss et al. 2021) aim to reduce uncertainty regarding the location of high-performing areas within the search space, as measured in terms of differential entropy. Penalisation-based methods (Ginsbourger, Le Riche, and Carraro 2010; Azimi, Fern, and Fern 2010; Desautels, Krause, and Burdick 2014; González et al. 2016; Alvi et al. 2019) apply penalties to either an acquisition function or a surrogate model to prevent the selection of solutions that have previously been chosen.

Recent advances in batch methods involve multi-objective optimisation by optimising multiple acquisition functions simultaneously. One approach (De Ath et al. 2020; De Ath, Everson, and Fieldsend 2021) selects only one solution from the Pareto front of μ and σ, with the remaining $q-1$ solutions chosen by other methods like Thompson sampling. Another approach selects all batch solutions from the Pareto front. For example, Bischl et al. (2014) considers multiple objectives, including EI, uncertainty, and distance metrics. Feng et al. (2015) and Grobler, Kok, and Wilke (2017) treat the two terms in EI as separate objectives. Lyu et al. (2018) and Chen, Luo, and Wang (2022); Chen et al. (2023) consider PI, EI, and LCB, selecting solutions randomly from the Pareto front or using strategies like k-means clustering. However, different acquisition functions may exhibit similar behaviours, resulting in similar solutions being selected together.

It is worth noting that like our method, several studies consider multi-objective optimisation for quality and uncertainty (Gupta et al. 2018; Binois, Collier, and Ozik 2021). In Gupta et al. (2018), the first point is selected based on LCB, with the remaining $q-1$ points chosen randomly. In Binois, Collier, and Ozik (2021), a hypervolume-based metric selects q points. However, they do it in a static manner, i.e., selecting q points simultaneously without considering the change of the uncertainty after each solution is selected. This may lead to selecting multiple good trade-off solutions in the same area (since closely-located solutions may have similar quality and uncertainty), reducing the algorithm's ability of exploration over the search space.

Multi-Objective Optimisation

Multi-objective optimisation involves finding optimal solutions for problems with multiple objectives. Mathematically, it is represented as: $\min F(x) = (f_1(x), ..., f_n(x))^T$, where n represents the number of objectives, and each f_i is the ith objective function. A solution x_1 dominates another solution x_2, denoted $x_1 \prec x_2$, if $f_i(x_1) \leq f_i(x_2)$ for all $i \in \{1, \ldots, n\}$, and there is at least one $j \in \{1, \ldots, n\}$ for which $f_j(x_1) < f_j(x_2)$. A solution $x_1 \in \chi$ is Pareto optimal if no other solution dominates it. The set of all Pareto optimal solutions is the Pareto set (PS), and its image in the objective space is the Pareto front (PF).

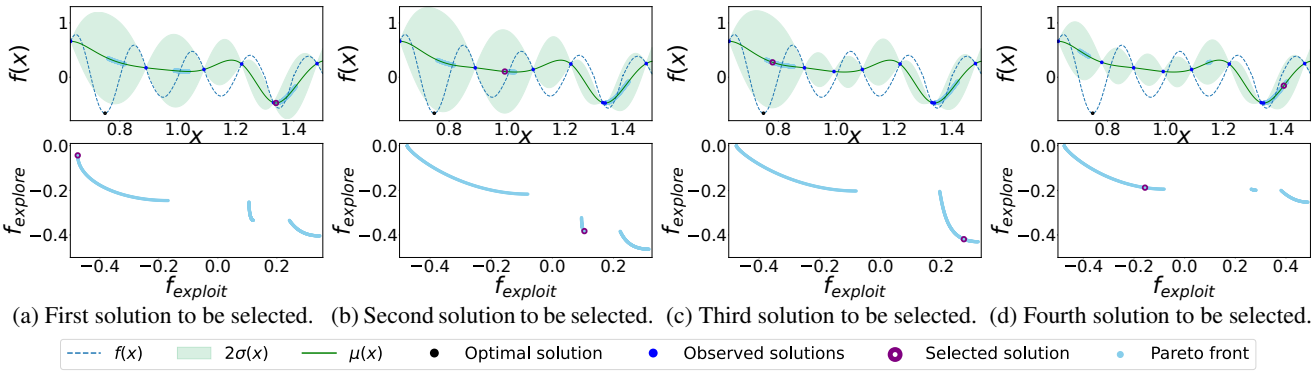
The Proposed POEE

Basic Idea

In BBO, it is critical to select a number of solutions in a batch that represent good, yet different, trade-offs between exploitation and exploration. It may not be beneficial to select a batch of high-quality solutions that are close to each other. Given this, when treating exploitation (the posterior mean) and exploration (the posterior uncertainty) as two objectives in BBO, a straightforward idea is to consider a group of diverse weight combinations that represent different trade-offs between exploitation and exploration. However, a problem with this approach is that some weight combinations (e.g., those overly in favour of the exploration) may not be helpful. For example, a weight combination of (0.2, 0.8) sees exploration as four times more important than exploitation. Considering such a weight combination is less likely to find promising solutions as the search may end up in an area with high uncertainty but low quality. In this paper, we take a different approach instead; we stick to a weight combination around the middle, but the Pareto front itself will change after one solution is picked since the uncertainty of the area around that solution is changed. In this case, we can have a number of solutions with good trade-offs between exploitation and exploration in different areas. In the following section, we will describe our method in detail.

The Proposed Batch Method

We first select the most exploitative solution from the Pareto front with respect to the exploitation (the posterior mean) and exploration (the posterior uncertainty) objectives. This is beneficial to find the solution with the highest quality provided by the Gaussian process model, as shown and practised in (De Ath et al. 2020; De Ath, Everson, and Fieldsend 2021). For the rest of $q-1$ trade-off solutions, we employ TOPSIS (Hwang and Yoon 1981), a multi-criteria decision making technique, to iteratively select the best trade-off solution from the exploitation-exploration Pareto front among a number of solutions based on weights. TOPSIS considers a solution that under specific weights is the closest to the ideal point and farthest to the negative-ideal point of the Pareto front (a detailed explanation of TOPSIS can be found in the supplementary material). It has been used to select the trade-off solution in sequential BO (Jiang and Li 2025). The



(a) First solution to be selected. (b) Second solution to be selected. (c) Third solution to be selected. (d) Fourth solution to be selected.

Figure 2: The example of Figure 1, in which the proposed method POEE is used to select a batch of solutions based on the dynamically updated Pareto front. In the top panel, the objective function is represented by a blue dashed line, approximated by the Gaussian Process regression. The mean $\mu(x)$ is depicted in dark green, while twice the posterior standard deviation $\sigma(x)$ is illustrated by the light green shaded area. Evaluated solutions are marked in blue. In the bottom panel, the cyan solutions are the Pareto front of objectives $f_{exploit}$ ($\mu(x)$) and $f_{explore}$ (-$\sigma(x)$) (to be minimised). The selected solution is marked in purple. During the batch selection process, the first solution chosen is the most exploitative solution, followed by the trade-off solutions identified by TOPSIS for the remaining $q-1$ solutions iteratively based on the dynamic Pareto front.

computational cost of TOPSIS is negligible ($O(m)$), where m is the number of the candidate solutions. After we select one solution, the uncertainty of other solutions is changed. We then update the Pareto front and again use TOPSIS to select the next solution under the same weights. This dynamically updated Pareto front allows the exploration on different trade-off areas. Here, the weight combination of the two objectives is set to (0.4, 0.6); its sensitivity study will be given in the experiment section.

Note that there exist some studies in BBO that work on solution diversity, such as selecting a set of diverse solutions by Determinantal Point Process (Kathuria, Deshpande, and Kohli 2016; Wang et al. 2017b) and defining diversity metrics (Konakovic Lukovic, Tian, and Matusik 2020; Ahmadianshalchi, Belakaria, and Doppa 2024). Unlike these methods that consider multiple points from a set simultaneously, our method works on only one trade-off solution at a time (i.e., from one Pareto front).

To help further understand the proposed method, Figure 2 illustrates its process of selecting a batch of solutions based on the example of Figure 1. As can be seen in the figure, we first select the solution with the highest quality from the quality-uncertainty Pareto front (Figure 2(a)). After that, the Pareto front is changed. We then sequentially select three good trade-off solutions from the dynamically updated Pareto front sitting at different trade-off areas within the search space.

Algorithm Framework

Algorithm 1 delineates the procedure of our proposed framework POEE. The algorithm commences with inputs including q: a batch size; MOEA: a multi-objective evolutionary algorithm; $f_{exploit}(x)$ and $f_{explore}(x)$: exploitation and exploration objectives; and $W = \{w_{exploit}, w_{exploit}\}$: weights of the two objectives used in TOPSIS. The output of the algorithm is the selected q solutions. At first, a multi-objective evolutionary algorithm[2] is used to optimise $\mu(x)$ and $-\sigma(x)$ to obtain a set of solutions with good quality and uncertainty (stored in an archive). At each iteration, the approximate Pareto front $\mathcal{P}$ is identified (step 4). The selection process involves choosing the most exploitative solution (for the first solution) and employing the TOPSIS technique to determine the remaining $q-1$ solutions. Following the selection of each solution, we update the objective function values, i.e., $-\sigma(x)$, in the $Archive$ via Equation (3) (step 12).

As can be seen from Equation (3), the update of uncertainty depends on the kernel function k and dataset X. To be specific, we use the same kernel hyperparameters of the kernel function in a batch. The dataset X consists of two types of points. The first type is concerned with the evaluated points and the second type is concerned with the selected points for evaluation. In the batch selection, the set of points of the first type does not change, while the set of points of the second type changes once a point is selected, hence resulting in the change of the uncertainty.

Experimental Setting

We consider 14 well-known synthetic and real-world problems (e.g., robot pushing (Wang and Jegelka 2017)), following the practice in the related papers (De Ath et al. 2021; De Ath, Everson, and Fieldsend 2021). The detailed descriptions are provided in the supplementary material.

A zero-mean Gaussian process surrogate model with a Matérn 5/2 kernel was used in all the experiments. The kernel was selected due to its wide usage and recommended use for modelling realistic functions (Snoek, Larochelle, and Adams 2012; Alghamdi et al. 2020). The models were initially trained on $2d$ initial solutions generated by the Latin hypercube sampling (Stein 1987), with each optimisation run repeated 30 times with different initialisations. The same

[2]Here, the well-known NSGA-II (Deb et al. 2002) is used.

Algorithm 1: POEE

Input: q: batch size; MOEA: multi-objective evolutionary algorithm; $f_{exploit}(x)$ and $f_{explore}(x)$: exploitation and exploration objectives; $W = \{w_{exploit}, w_{exploit}\}$: weights of the two objectives used in TOPSIS.

Output: P: a batch of solutions.

$P \leftarrow \emptyset$

$Archive \leftarrow MOEA(f_{exploit}(x), f_{explore}(x))$

`// Return all solutions in the optimisation process`

for $i \leftarrow 1$ *to* q **do**

  $\{x_i\}_{i=1}^{m} \leftarrow find\_Pareto\_front(Archive)$

  if $i = 1$ **then**

    $x_p \leftarrow \arg\min_{x \in \{x_i\}_{i=1}^{m}} f_{exploit}(x)$

  else

    $\mathbf{F}_{exploit} \leftarrow (f_{exploit}(x_1), ..., f_{exploit}(x_m))^T$

    $\mathbf{F}_{explore} \leftarrow (f_{explore}(x_1), ..., f_{explore}(x_m))^T$

    $x_p \leftarrow TOPSIS(W, (\mathbf{F}_{exploit}\ \mathbf{F}_{explore})_{m \times 2})$

  $P \leftarrow P \cup \{x_p\}$

  Update objective function values in the $Archive$ via Equation (3)

sets of initial batch solutions were same across all methods to enable statistical comparison. At each iteration in BO, before the selection of batch solutions, the hyperparameters of the Gaussian process were optimised by maximising the log likelihood via L-BFGS-B (Zhu et al. 1997).

To evaluate the proposed POEE, we consider 12 well-established batch methods. Among them, there are one baseline method (i.e., random search (RS)), three penalisation-based methods (i.e., Local Penalisation (LP) (González et al. 2016), PLAyBOOK (Alvi et al. 2019), and Kriging Believer (KB) (Ginsbourger, Le Riche, and Carraro 2010)), two greed-based batch methods (i.e., ϵ-shotgun (De Ath et al. 2020) and AEGiS (De Ath, Everson, and Fieldsend 2021)), and one information-based method (i.e., GIBBON (Moss et al. 2021)). We also consider the stochastic method (i.e., standard Thompson Sampling (TS) (Kandasamy et al. 2018)), and the qEI method (Ginsbourger, Le Riche, and Carraro 2008; Balandat et al. 2020) which is designed to jointly estimate the locations of the q batch members. Additionally, the trust regions-based method is considered (i.e., TuRBO (Eriksson et al. 2019)). Lastly, we consider the Pareto optimisation-based methods (i.e., MACE (Lyu et al. 2018) and Gupta (Gupta et al. 2018)). Note that we currently used standard acquisition functions without logarithm. However, using the logarithm could mitigate vanishing gradient issues (especially for improvement-based methods) and improve numerical stability (Ament et al. 2023).

We implemented qEI, GIBBON, and TuRBO through BoTorch (Balandat et al. 2020) and the other methods were implemented by GPy (suggested by (De Ath et al. 2020)). We followed guidelines of the original papers in parameter settings. The LP, PLAyBOOK, and KB methods all employed the well-known and effective EI acquisition function. For ϵS-PF, we set $\epsilon = 0.1$, and for AEGiS, ϵ was determined as $\min(2/\sqrt{d}, 1)$ with $\epsilon t = \epsilon p = \epsilon/2$. For TuRBO, Thompson sampling is used as the acquisition function. The MACE optimised PI, EI, and LCB using NSGA-II, with the LCB parameter $\sqrt{\beta_t}$ defined as $\sqrt{v \log(t^{d/2+2}\pi^2/3\delta)}$, where $v = 0.5$ and $\delta = 0.05$. In the proposed POEE method, we assigned the weights for exploitation and exploration as $w_{exploit} = 0.4$ and $w_{explore} = 0.6$, respectively.

For the methods utilising NSGA-II, including POEE, ϵS-PF, AEGiS, MACE, and Gupta, parameters were set to commonly accepted values: a population size of 100, crossover and mutation probabilities of 1.0 and d^{-1} respectively, and distribution indices of 20 for both crossover and mutation. For each solution selected in LP and PLAyBOOK, we followed the authors' guideline (Alvi et al. 2019) and uniformly sampled the acquisition function at 3000 locations, selecting the best solution after locally optimising (with L-BFGS-B) the best 5. For the other methods (excluding LP and PLAyBOOK), a maximum budget of $10000d$ acquisition function evaluations was used, as suggested by De Ath et al. (2020). For methods implemented using BoTorch (i.e., qEI, GIBBON, and TuRBO), the L-BFGS-B algorithm is used for optimising aquisition functions. For KB, TS, ϵ-PF, and AEGiS, optimisation for functions with $d = 1$ utilised the L-BFGS-B algorithm, while for $d \geq 2$, the CMA-ES (Hansen 2009) algorithm was employed, following the standard bi-population strategy and allowing up to 9 restarts (suggested by De Ath et al. (2020)).

Experimental Results

Comparison with Well-Established Methods

The methods were evaluated on the 14 synthetic and practical problems with batch sizes $q \in \{5, 10, 20\}$ and a fixed budget of 300 function evaluations. Convergence plots and tabulated results for the 14 functions are available in the supplementary material. Table 1 presents the results (mean and standard deviation) of the regrets obtained for a batch size of $q = 5$ and the statistical results based on the Wilcoxon signed-rank test (Sidney 1957). Additionally, the comparative results between POEE and the peer algorithms are summarised in Tables 2.

As can be seen in Table 1, POEE shows overall better performance than RS, qEI, LP, PLAyBOOK, KB, TS, ϵS-PF, AEGiS, GIBBON, MACE, Gupta, and TuRBO. Particularly, for the problem WangFreitas which is 1D, POEE, RS, qEI, and AEGiS demonstrate their ability to find the global optimum, while the algorithms LP, PLAyBOOK, KB, ϵS-PF, MACE, Gupta, and TuRBO fail. For the 2D optimisation problems, POEE obtains statistically equivalent performance or the lowest mean regrets, apart from the test problem Branin. For all the 10D problems, our method outperforms the other methods significantly. For practical problems, POEE achieves the best mean regret on Push4 and presents statistically equivalent performance to the best acquisition function, KB, on push8.

Interestingly, among the three penalisation-based methods using EI, i.e., KB, LP and PLAyBOOK, which either penalise the acquisition function or the surrogate model, the

Method	WangFreitas (1)		BraninForrester (2)		Branin (2)		Eggholder (2)		GoldsteinPrice (2)		SixHumpCamel (2)		Hartmann6 (6)	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
POEE	6.08e-8	1.2e-7	**5.98e-8**	**9.6e-8**	1.86e-6	1.8e-6	3.69e+1	3.0e+1	2.36e-1	8.8e-1	**6.29e-8**	**5.9e-8**	2.79e-2	5.1e-2
RS	7.05e-2	1.6e-1(−)	6.15e-1	7.8e-1(−)	1.98e-1	1.3e-1(−)	1.53e+2	7.3e+1(−)	5.35e+0	6.0e+0(−)	7.17e-2	6.3e-2(−)	9.55e-1	2.9e-1(−)
qEI	**3.71e-8**	**1.1e-7**(∼)	4.09e-6	7.7e-6(−)	2.55e-6	2.6e-6(∼)	8.66e+1	5.3e+1(−)	6.01e-1	5.7e-1(−)	9.90e-6	9.7e-6(−)	4.53e-2	7.1e-2(−)
LP	1.87e+0	5.0e-1(−)	3.51e-5	1.3e-4(−)	4.64e-6	9.3e-6(∼)	3.92e+1	3.3e+1(∼)	**5.76e-2**	**9.0e-2**(∼)	6.22e-6	2.3e-5(−)	2.75e-2	4.8e-2(+)
PLAyBOOK	1.80e+0	6.0e-1(−)	6.15e-5	2.0e-4(−)	6.60e-6	1.0e-5(−)	4.35e+1	3.2e+1(∼)	1.93e-1	2.6e-1(∼)	3.61e-6	5.7e-6(−)	4.08e-2	5.7e-2(−)
KB	1.87e+0	5.0e-1(−)	4.43e-7	2.0e-6(∼)	**7.19e-7**	**9.8e-7**(+)	**2.86e+1**	**2.5e+1**(∼)	4.22e-1	1.2e+0(∼)	1.22e-6	2.4e-6(−)	**2.01e-2**	**4.5e-2**(+)
TS	1.87e+0	5.0e-1(−)	9.24e-5	8.1e-5(−)	4.73e-5	2.4e-5(−)	4.87e+1	2.0e+1(∼)	4.58e-1	3.5e-1(−)	6.39e-5	2.4e-5(−)	7.00e-2	3.8e-2(−)
ϵS-PF	1.67e+0	7.5e-1(−)	9.20e-5	1.7e-4(−)	6.44e-5	1.7e-4(−)	2.05e+2	1.2e+2(−)	7.35e+0	1.1e+1(−)	3.95e-5	1.2e-4(−)	3.71e-2	5.7e-2(−)
AEGiS	6.79e-7	1.5e-6(−)	6.01e-4	3.4e-4(−)	3.84e-4	4.9e-4(−)	2.94e+1	2.4e+1(∼)	1.59e+0	1.1e+0(−)	2.05e-4	2.1e-4(−)	2.25e-2	4.9e-2(+)
GIBBON	8.00e-1	9.8e-1(−)	3.70e-4	5.6e-4(−)	8.10e-4	1.1e-3(−)	6.70e+1	5.7e+1(−)	3.04e+0	3.6e+0(−)	1.63e-3	2.6e-3(−)	2.58e-2	4.9e-2(+)
MACE	1.87e+0	5.0e-1(−)	1.70e-5	5.6e-5(−)	2.13e-6	4.1e-6(∼)	4.77e+1	6.1e+1(∼)	4.43e-1	5.1e-1(−)	1.65e-7	2.9e-7(∼)	3.75e-2	6.1e-2(−)
Gupta	1.87e+0	5.0e-1(−)	3.05e-1	1.6e+0(−)	2.43e-5	7.4e-5(∼)	4.35e+1	4.9e+1(∼)	1.21e-1	1.4e-1(+)	9.34e-7	4.1e-6(∼)	3.26e-2	5.4e-2(−)
TuRBO	1.87e+0	5.0e-1(−)	1.06e+1	1.9e+1(−)	2.39e-4	4.4e-4(−)	4.11e+2	1.9e+2(−)	3.97e+1	1.5e+2(−)	6.09e-1	1.0e+0(−)	2.86e-1	3.9e-1(−)

Method	Ackley (2)		Griewank (2)		Ackley (10)		Griewank (10)		GSobol (10)		Push4 (4)		Push8 (8)	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
POEE	**1.12e-2**	**7.9e-3**	**1.21e-1**	**1.2e-1**	**3.06e+0**	**3.3e+0**	**7.19e-1**	**1.9e-1**	**6.35e+2**	**6.2e+2**	**1.50e-2**	**1.9e-2**	1.48e+0	1.8e+0
RS	5.64e+0	2.3e+0(−)	1.19e+0	4.4e-1(−)	1.85e+1	1.2e+0(−)	8.84e+1	1.9e+1(−)	3.39e+3	2.6e+3(−)	5.08e-1	2.7e-1(−)	3.78e+0	2.0e+0(−)
qEI	3.99e+0	2.3e+0(−)	9.02e-1	4.8e-1(−)	1.88e+1	8.0e-1(−)	8.44e+1	2.2e+1(−)	2.23e+3	2.6e+3(−)	1.55e-1	9.7e-2(−)	1.70e+0	1.3e+0(∼)
LP	5.33e-1	1.6e+0(−)	3.95e-1	1.7e-1(−)	1.26e+1	2.8e+0(−)	3.36e+0	9.6e-1(−)	1.07e+3	1.6e+3(∼)	1.10e-1	8.9e-2(−)	1.92e+0	1.6e+0(∼)
PLAyBOOK	5.58e-1	2.2e+0(−)	4.79e-1	2.3e-1(−)	1.45e+1	2.8e+0(−)	3.44e+0	1.0e+0(−)	1.34e+3	1.1e+3(−)	1.57e-1	2.2e-1(−)	2.11e+0	1.9e+0(∼)
KB	3.00e-1	2.7e-1(−)	3.55e-1	1.8e-1(−)	9.76e+0	7.0e+0(−)	9.96e-1	4.4e-2(−)	1.93e+3	2.4e+3(−)	1.29e-1	1.1e-1(−)	**1.16e+0**	**9.7e-1**(∼)
TS	1.63e-1	3.0e-2(−)	3.25e-1	1.5e-1(−)	1.32e+1	7.3e-1(−)	1.21e+1	8.8e-1(−)	5.05e+3	4.8e+3(−)	3.44e-1	1.7e-1(−)	2.69e+0	1.6e+0(−)
ϵS-PF	1.55e+0	2.8e+0(∼)	1.35e-1	1.6e-1(∼)	1.68e+1	2.9e+0(−)	8.71e-1	1.5e-1(−)	1.52e+3	2.5e+3(∼)	1.97e-2	1.5e-2(∼)	2.04e+0	2.0e+0(∼)
AEGiS	5.46e-1	6.9e-1(−)	3.61e-1	1.8e-1(−)	1.30e+1	3.6e+0(−)	1.76e+0	2.7e-1(−)	2.88e+3	3.6e+3(−)	2.80e-1	1.3e-1(−)	1.89e+0	1.6e+0(∼)
GIBBON	2.55e+0	4.4e+0(−)	3.70e-1	2.5e-1(−)	1.86e+1	9.6e-1(−)	8.61e+1	1.8e+1(−)	2.67e+3	2.9e+3(−)	1.25e-1	1.6e-1(−)	2.32e+0	1.8e+0(−)
MACE	2.65e-1	9.6e-1(−)	6.43e-1	1.1e+0(−)	1.66e+1	2.1e+0(−)	8.06e-1	1.9e-1(∼)	1.24e+3	3.8e+3(∼)	1.54e-1	2.5e-1(−)	2.48e+0	2.1e+0(−)
Gupta	3.55e-1	1.1e+0(−)	4.48e-1	2.3e-1(−)	1.36e+1	2.7e+0(−)	3.40e+0	9.6e-1(−)	1.54e+3	1.1e+3(−)	1.60e-1	2.2e-1(−)	1.87e+0	1.5e+0(∼)
TuRBO	1.63e+1	4.2e+0(−)	2.12e+0	2.2e+0(−)	1.92e+1	4.9e-1(−)	2.91e+0	3.0e+0(−)	1.28e+3	2.0e+3(∼)	2.97e-1	8.1e-1(∼)	2.97e+0	2.5e+0(−)

Table 1: The performance (mean and standard deviation) of POEE and the other 12 methods were evaluated across the 14 problems with a batch size of $q = 5$. The method demonstrating the best mean performance is denoted in bold. The symbols "+", "∼", and "−" indicate that the method is statistically better than, equivalent to, and worse than POEE respectively.

	RS	qEI	LP	PLAyBOOK	KB	TS	ϵS-PF	AEGiS	GIBBON	MACE	Gupta	TuRBO
POEE	14/ 0/ 0	11/ 3/ 0	8/ 5/ 1	11/ 3/ 0	8/ 4/ 2	13/ 1/ 0	9/ 5/ 0	11/ 2/ 1	13/ 0/ 1	9/ 5/ 0	9/ 4/ 1	12/ 2/ 0

Table 2: The summary of statistical results, as detailed in Table 1. Here, the left, median, and right numbers are the counts of test problems where the POEE was statistically superior, equivalent, or inferior to the peer algorithm, respectively.

older KB performed better than the newer LP and PLAyBOOK. The proposed POEE, which penalises the quality-uncertainty Pareto front, performed better than KB, LP and PLAyBOOK. As might be expected, MACE's performance declines with larger batch sizes (q). This decline may result from the selection of more similar solutions within a batch as q increases.

Figure 3 shows convergence trajectories of the 13 algorithms on the two problems. As can be seen, in the test problem Push4, POEE performs significantly better than its peers. Regarding Griewank (2D), POEE does not perform best in the early optimisation stage, but it ultimately achieves the lowest mean regret.

Additionally, we considered the observational noise ($\sigma_\epsilon = 0.1$ (Wang and Jin 2023)) on the 14 problems using a batch size of $q = 5$. Table 3 summarises the statistical results. The results show that POEE generally statistically outperforms most peer algorithms across the majority of the test problems, particularly against LP, PLAyBOOK, and AEGiS.

We also conducted the ablation study to evaluate the contributions of three key components of POEE: 1) the dynamically updated Pareto front, 2) the initial selection of the most exploitative solution, and 3) the use of TOPSIS. Due to page limit, the results are given in the supplementary text. In general, each component of our POEE method plays an important role in striking a good balance between exploitation and exploration. Regarding the computational cost (see the supplementary), POEE, though slower than most of the others, is acceptable (172s on a 10D problem with the batch size 10), compared to the usual expensive evaluation of a solution involved in BO.

Sensitivity Analysis

To better balance exploitation and exploration, we conducted a sensitivity analysis with five weight combinations: 1) 0.2/0.8, 2) 0.4/0.6 (used in POEE), 3) 0.5/0.5, 4) 0.6/0.4, and 5) 0.8/0.2. These combinations cover the spectrum from exploration-preferred to exploitation-preferred approaches.

Figure 4 summarises the sensitivity analysis results. Detailed convergence plots and tabulated results are available in the supplementary material. As can be seen in the figure, the methods heavily skewed towards either exploitation (POEE_{82}) or exploration (POEE_{28}) generally perform worse regardless of batch size. However, our findings suggest a slight preference for exploration over exploitation, as POEE_{28} marginally outperforms POEE_{82}. This indicates that within the POEE framework, prioritising exploration may yield better outcomes than exploitation. Moderate con-

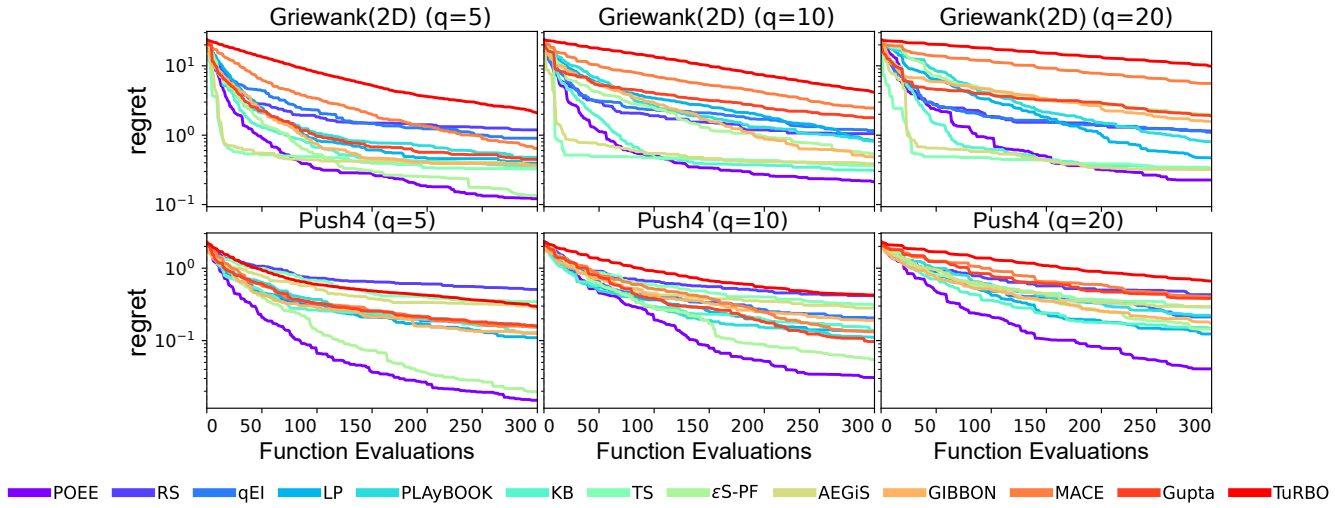


Figure 3: The convergence trajectories of the 13 methods throughout the optimisation process on the two problems. Each coloured line illustrates the mean difference between the true optimum and the best function value obtained over 30 runs.

	qEI	LP	PLAyBOOK	KB	TS	ϵS-PF	AEGiS	GIBBON	MACE	Gupta
POEE	7/ 5/ 2	12/ 2/ 0	13/ 1/ 0	4/ 7/ 3	7/ 6/ 1	9/ 3/ 2	11/ 3/ 0	8/ 5/ 1	7/ 4/ 3	11/ 1/ 2

Table 3: Result summary of the noisy experiments with noise level $\sigma_\epsilon = 0.1$ on the 14 problems using a batch size of $q = 5$. Here, the left, median, and right numbers are the counts of test problems where the POEE was statistically superior, equivalent, or inferior to the peer algorithm, respectively.

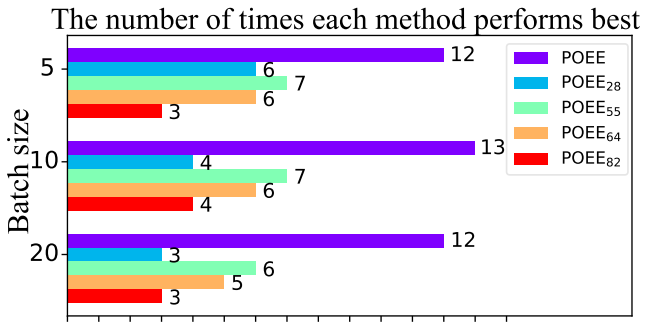


Figure 4: Result summary of the sensitivity analysis, where the bars correspond to the number of times that a method is statistically best across the 14 synthetic and practical functions with $q \in \{5, 10, 20\}$.

figurations, i.e., POEE (POEE$_{46}$), POEE$_{64}$ and POEE$_{55}$, perform generally better. This confirms the need for a good balance between exploitation and exploration. Among all the configurations, POEE (POEE$_{46}$) emerges as the most effective one, achieving the best result on more problems.

It is worth mentioning that although POEE generally works well with the weights (0.4, 0.6), different problem functions may prefer different weights. Adapting weights based on search feedback on a given problem may further improve the performance of the proposed algorithm.

Discussion

In the proposed POEE, we considered 10,000d solutions generated by NSGA-II at the beginning of each batch. After the update of the uncertainty objective, we did not rerun NSGA-II to optimise the updated problem. It is worth stating that re-running the algorithm could be beneficial, particularly if setting fewer solutions (e.g., 1,000d) per run. We will explore it in our future study.

Like many other methods which use the standard GP model, our method may face scalability issues with higher dimensions. A possible way to mitigate it is incorporating techniques like principal component analysis (Antonov et al. 2022) or trust regions (Eriksson et al. 2019). This is worth further investigation.

Conclusion

Balancing exploration and exploitation is a challenging task in BBO. The proposed method POEE attempted to address this via seeking good trade-offs between them. By considering a dynamically updated Pareto front with respect to exploitation and exploration, POEE is able to identify a batch of solutions that represent different trade-offs between the posterior mean and uncertainty.

In this work, we considered 14 test problems, including 2 real-world instances. In future study, we will apply our method to more real-world problems, such as hyperparameter tuning (Kathuria, Deshpande, and Kohli 2016), walker function (Wang et al. 2017b), and computational fluid dynamics optimisation problems (De Ath et al. 2021).

References

Ahmadianshalchi, A.; Belakaria, S.; and Doppa, J. R. 2024. Pareto Front-Diverse Batch Multi-Objective Bayesian Optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 10784–10794.

Alghamdi, A. S.; Polat, K.; Alghoson, A.; Alshdadi, A. A.; and Abd El-Latif, A. A. 2020. Gaussian process regression (GPR) based non-invasive continuous blood pressure prediction method from cuff oscillometric signals. *Applied Acoustics*, 164: 107256.

Alvi, A.; Ru, B.; Calliess, J.-P.; Roberts, S.; and Osborne, M. A. 2019. Asynchronous Batch Bayesian Optimisation with Improved Local Penalisation. In *Proceedings of the 36th International Conference on Machine Learning*, 253–262. PMLR.

Ament, S.; Daulton, S.; Eriksson, D.; Balandat, M.; and Bakshy, E. 2023. Unexpected Improvements to Expected Improvement for Bayesian Optimization. In *Advances in Neural Information Processing Systems*, volume 36, 20577–20612. Curran Associates, Inc.

Antonov, K.; Raponi, E.; Wang, H.; and Doerr, C. 2022. High dimensional Bayesian optimization with kernel principal component analysis. In *International Conference on Parallel Problem Solving from Nature*, 118–131. Springer.

Azimi, J.; Fern, A.; and Fern, X. 2010. Batch Bayesian Optimization via Simulation Matching. In *Advances in Neural Information Processing Systems*, volume 23, 109–117. Curran Associates, Inc.

Balandat, M.; Karrer, B.; Jiang, D.; Daulton, S.; Letham, B.; Wilson, A. G.; and Bakshy, E. 2020. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems*, volume 33, 21524–21538. Curran Associates, Inc.

Binois, M.; Collier, N.; and Ozik, J. 2021. A portfolio approach to massively parallel Bayesian optimization. *arXiv preprint arXiv:2110.09334*.

Bischl, B.; Wessing, S.; Bauer, N.; Friedrichs, K.; and Weihs, C. 2014. MOI-MBO: multiobjective infill for parallel model-based optimization. In *International Conference on Learning and Intelligent Optimization*, 173–186. Springer.

Chen, J.; Luo, F.; Li, G.; and Wang, Z. 2023. Batch Bayesian optimization with adaptive batch acquisition functions via multi-objective optimization. *Swarm and Evolutionary Computation*, 79: 101293.

Chen, J.; Luo, F.; and Wang, Z. 2022. Dynamic multi-objective ensemble of acquisition functions in batch Bayesian optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 479–482.

Chevalier, C.; and Ginsbourger, D. 2013. Fast computation of the multi-points expected improvement with applications in batch selection. In *International Conference on Learning and Intelligent Optimization*, 59–69. Springer.

Chugh, T. 2020. Scalarizing functions in Bayesian multiobjective optimization. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, 1–8. IEEE.

Daxberger, E. A.; and Low, B. K. H. 2017. Distributed batch Gaussian process optimization. In *Proceedings of the 34th International Conference on Machine Learning*, 951–960. PMLR.

De Ath, G.; Everson, R. M.; and Fieldsend, J. E. 2021. Asynchronous ε-Greedy Bayesian Optimisation. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161, 578–588. PMLR.

De Ath, G.; Everson, R. M.; Fieldsend, J. E.; and Rahat, A. A. 2020. ε-shotgun: ε-greedy batch Bayesian optimisation. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, 787–795. Association for Computing Machinery.

De Ath, G.; Everson, R. M.; Rahat, A. A.; and Fieldsend, J. E. 2021. Greed is good: Exploration and exploitation trade-offs in Bayesian optimisation. *ACM Transactions on Evolutionary Learning and Optimization*, 1(1): 1–22.

Deb, K.; Pratap, A.; Agarwal, S.; and Meyarivan, T. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2): 182–197.

Desautels, T.; Krause, A.; and Burdick, J. W. 2014. Parallelizing exploration-exploitation tradeoffs in Gaussian process bandit optimization. *Journal of Machine Learning Research*, 15(119): 4053–4103.

Eriksson, D.; Pearce, M.; Gardner, J.; Turner, R. D.; and Poloczek, M. 2019. Scalable global optimization via local Bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 32, 5496–5507. Curran Associates, Inc.

Feng, Z.; Zhang, Q.; Zhang, Q.; Tang, Q.; Yang, T.; and Ma, Y. 2015. A multiobjective optimization based framework to balance the global exploration and local exploitation in expensive optimization. *Journal of Global Optimization*, 61(4): 677–694.

Garnett, R. 2023. *Bayesian optimization*. Cambridge University Press.

Ginsbourger, D.; Le Riche, R.; and Carraro, L. 2008. A multi-points criterion for deterministic parallel global optimization based on Gaussian processes. Technical report.

Ginsbourger, D.; Le Riche, R.; and Carraro, L. 2010. Kriging is well-suited to parallelize optimization. In *Computational Intelligence in Expensive Optimization Problems*, volume 2, 131–162. Springer.

González, J.; Dai, Z.; Hennig, P.; and Lawrence, N. 2016. Batch Bayesian optimization via local penalization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51, 648–657. PMLR.

Grobler, C.; Kok, S.; and Wilke, D. N. 2017. Simple intuitive multi-objective parallelization of efficient global optimization: simple-ego. In *World Congress of Structural and Multidisciplinary Optimisation*, 205–220. Springer.

Gupta, S.; Shilton, A.; Rana, S.; and Venkatesh, S. 2018. Exploiting strategy-space diversity for batch Bayesian optimization. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, 538–547. PMLR.

Hansen, N. 2009. Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, 2389–2396.

Huang, X.; Song, L.; Xue, K.; and Qian, C. 2024. Stochastic Bayesian optimization with unknown continuous context distribution via kernel density estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 12635–12643.

Hunt, N. 2020. *Batch Bayesian optimization*. Ph.D. thesis, Massachusetts Institute of Technology.

Hwang, C.-L.; and Yoon, K. 1981. *Multiple attribute decision making and applications*. Springer.

Jiang, C.; and Li, M. 2025. Multi-objectivising acquisition functions in Bayesian optimisation. *ACM Transactions on Evolutionary Learning and Optimization*, to appear.

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

Kandasamy, K.; Krishnamurthy, A.; Schneider, J.; and Póczos, B. 2018. Parallelised Bayesian optimisation via Thompson sampling. In *International Conference on Artificial Intelligence and Statistics*, 133–142. PMLR.

Kathuria, T.; Deshpande, A.; and Kohli, P. 2016. Batched Gaussian Process Bandit Optimization via Determinantal Point Processes. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Konakovic Lukovic, M.; Tian, Y.; and Matusik, W. 2020. Diversity-Guided Multi-Objective Bayesian Optimization With Batch Evaluations. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 17708–17720. Curran Associates, Inc.

Kushner, H. J. 1964. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal Basic Engineering*, 86(1): 97–106.

Lai, T. L.; and Robbins, H. 1985. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1): 4–22.

Lyu, W.; Yang, F.; Yan, C.; Zhou, D.; and Zeng, X. 2018. Batch Bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, 3306–3314. PMLR.

MacKay, D. J. 1998. Introduction to Gaussian processes. *NATO ASI Series F Computer and Systems Sciences*, 168: 133–166.

Mockus, J.; Tiesis, V.; and Zilinskas, A. 1978. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 2(2): 117–129.

Moss, H. B.; Leslie, D. S.; Gonzalez, J.; and Rayson, P. 2021. Gibbon: General-purpose information-based Bayesian optimisation. *Journal of Machine Learning Research*, 22(235): 1–49.

Santoni, M. L.; Raponi, E.; De Leone, R.; and Doerr, C. 2024. Comparison of High-Dimensional Bayesian Optimization Algorithms on BBOB. *ACM Transactions on Evolutionary Learning and Optimisation*, 4(3): 1–33.

Schonlau, M. 1997. *Computer experiments and global optimization*. Ph.D. thesis, University of Waterloo.

Shah, A.; and Ghahramani, Z. 2015. Parallel predictive entropy search for batch global optimization of expensive objective functions. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Sidney, S. 1957. Nonparametric statistics for the behavioral sciences. *The Journal of Nervous and Mental Disease*, 125(3): 497.

Snoek, J.; Larochelle, H.; and Adams, R. P. 2012. Practical Bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, 25.

Song, L.; Xue, K.; Huang, X.; and Qian, C. 2022. Monte carlo tree search based variable selection for high dimensional bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 35, 28488–28501. Curran Associates, Inc.

Stein, M. 1987. Large sample properties of simulations using Latin hypercube sampling. *Technometrics*, 29(2): 143–151.

Wang, H.; van Stein, B.; Emmerich, M.; and Back, T. 2017a. A new acquisition function for Bayesian optimization based on the moment-generating function. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 507–512. IEEE.

Wang, X.; and Jin, Y. 2023. Personalized Bayesian optimization for noisy problems. *Complex & Intelligent Systems*, 9(5): 5745–5760.

Wang, X.; Jin, Y.; Schmitt, S.; and Olhofer, M. 2023. Recent advances in Bayesian optimization. *ACM Computing Surveys*, 55(13s): 1–36.

Wang, Z.; and Jegelka, S. 2017. Max-value entropy search for efficient Bayesian optimization. In *Proceedings of the 34th International Conference on Machine Learning*, 3627–3635. PMLR.

Wang, Z.; Li, C.; Jegelka, S.; and Kohli, P. 2017b. Batched High-dimensional Bayesian Optimization via Structural Kernel Learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, 3656–3664. PMLR.

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

Wu, J.; and Frazier, P. 2016. The parallel knowledge gradient method for batch Bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 29, 3126–3134. Curran Associates, Inc.

Zhu, C.; Byrd, R. H.; Lu, P.; and Nocedal, J. 1997. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software*, 23(4): 550–560.