Proceedings of the 2023 Winter Simulation Conference
C. G. Corlu, S. R. Hunter, H. Lam, B. S. Onggo, J. Shortle, and B. Biller, eds.

HYPERPARAMETER ADAPTIVE SEARCH FOR SURROGATE OPTIMIZATION: A SELF-ADJUSTING APPROACH

Nazanin Nezami
Hadis Anahideh

Department of Mechanical and Industrial Engineering
University of Illinois at Chicago
842 W Taylor St
Chicago, IL 60607, USA

ABSTRACT

Surrogate Optimization (SO) algorithms have shown promise for optimizing expensive black-box functions. However, their performance is heavily influenced by hyperparameters related to sampling and surrogate fitting, which poses a challenge to their widespread adoption. We investigate the impact of hyperparameters on various SO algorithms and propose a Hyperparameter Adaptive Search for SO (HASSO) approach. HASSO is not a hyperparameter tuning algorithm, but a generic self-adjusting SO algorithm that dynamically tunes its own hyperparameters while concurrently optimizing the primary objective function, without requiring additional evaluations. The aim is to improve the accessibility, effectiveness, and convergence speed of SO algorithms for practitioners. Our approach identifies and modifies the most influential hyperparameters specific to each problem and SO approach, reducing the need for manual tuning without significantly increasing the computational burden. Experimental results demonstrate the effectiveness of HASSO in enhancing the performance of various SO algorithms across different global optimization test problems.

1 INTRODUCTION

Optimizing expensive black-box functions is a crucial and challenging task that arises in various real-world applications, including engineering design (Chen et al. 2019), financial management (Jafar 2022), drug discovery (Pyzer-Knapp 2018), and complex algorithms (Feurer and Hutter 2019). In such cases, evaluating the objective function is computationally expensive, and therefore, the evaluation budget is restricted. One of the most promising approaches to overcoming this challenge is to use SO algorithms (Anahideh et al. 2022). These algorithms involve constructing a cheap-to-evaluate model, known as a surrogate, that approximates the behavior of the black-box function. The surrogate model is then used to identify promising regions in the input space for further sampling and exploration.

Despite their effectiveness, SO algorithms heavily depend on multiple hyperparameters associated with sampling and surrogate fitting steps, leading to a significant impact on their performance across different problems. The selection of hyperparameters depends on several factors, such as the problem's characteristics, the complexity of the SO algorithm, and the available computational budget. The traditional approach for hyperparameter tuning involves manually adjusting hyperparameters using trial and error or grid search, either before or periodically during the optimization process. However, this approach can be tedious and time-consuming, especially when the number of hyperparameters and their search space is large. Hence, a self-adaptive SO approach that simultaneously tunes hyperparameters while optimizing the primary objective offers a promising solution to reduce the time and effort required to find global optima.

To overcome this challenge, a few studies in the field integrates the hyperparameter tuning process within the optimization framework, this approach automates the selection of hyperparameters and adapts them dynamically during the optimization process. Malkomes and Garnett (2018) proposed an automated Bayesian optimization approach that dynamically selects promising Gaussian Process (GP) models from a set of candidate models based on the model evidence metric (marginal likelihood), which measures how well the model explains the observed data. Likewise, Benjamins et al. (2022) demonstrated the benefits of dynamically selecting acquisition functions for Bayesian optimization design by training an acquisition function selector. The selector predicts which acquisition function will have the best performance and switches to that function. Existing research in SO has predominantly focused on tuning GP hyperparameters, overlooking hyperparameters in candidate generation and sampling steps. Additionally, most research has concentrated on individual hyperparameters, neglecting the joint tuning of multiple hyperparameters. To address these limitations, a more flexible approach is required that can handle diverse surrogate models and all other relevant hyperparameters. By broadening the scope of hyperparameter tuning and considering their collective adjustment within the SO framework, without the need for additional evaluations, we can significantly enhance the performance and versatility of SO algorithms.

Hyperparameter Adaptive Search for Surrogate Optimization (HASSO) is a novel approach that automates the selection of hyperparameters for SO algorithms while aiming for the primary global optima. HASSO incorporates a secondary search on the hyperparameter space alongside the primary search on the decision variables of the optimization problem. Drawing inspiration from Thompson sampling (Agrawal and Goyal 2012), HASSO dynamically tunes its own hyperparameters within each SO iteration. It treats each hyperparameter as an arm with a predefined range of values and adapts them based on the probability of improving the primary objective value. HASSO achieves self-adjustment without exhaustively searching for hyperparameters at every step, resulting in significant computational resources and time savings. Unlike existing methods (Malkomes and Garnett 2018), HASSO is not limited to surrogate model hyperparameters, making it versatile and adaptable to any SO algorithm. Experimental results validate the superior optimization performance of HASSO compared to several baselines, including fixed configurations, random updates, predefined rule-based updates, and grid search for hierarchical selection. HASSO outperforms these approaches across a diverse range of test problems, demonstrating its effectiveness and efficiency. By dynamically adapting hyperparameters within each iteration, HASSO achieves better convergence and exploration-exploitation balance, resulting in improved overall optimization performance.

The remainder of this paper is structured as follows: in Section 2, we provide background information on SO and discuss commonly used strategies. In Section 3, we demonstrate the impact of hyperparameters on the performance of various SO algorithms. Section 4 presents our proposed approach, and in Section 5, we report our experimental results. We discuss the implications of our findings and conclude in Section 6.

2 BACKGROUND

2.1 Surrogate Optimization

Surrogate optimization (SO) is an iterative method to optimize a black-box function f over a closed hypercube in $\mathbb{R}^d$. The hypercube is formed by the Cartesian product of $[x_1^L, x_1^U] \times [x_2^L, x_2^U] \times \cdots \times [x_d^L, x_d^U]$, where x_i^L and x_i^U are the lower and upper bounds for the $i-$th dimension of an input vector $\mathbf{x}$. SO starts with generating a set of initial points, $\mathscr{D} = \{\mathbf{x}^1, \ldots, \mathbf{x}^n\}$, using a Design of Experiments (DOE) method, and record their corresponding function values in $\mathscr{F} = \{f(\mathbf{x}^i) | \mathbf{x}^i \in \mathscr{D}\}$. Subsequently, a cheap-to-evaluate approximation model, $\hat{f}$, is fitted to the evaluated data points, $(\mathscr{D}, \mathscr{F})$. A sample generation scheme, such as random sampling, is used to discretize the solution space and generate the candidate set $\mathscr{C}$. Next, an acquisition function is utilized to select candidate points with optimum acquisition values, $\mathscr{S}$, to be evaluated using the expensive black-box system. That is, $\mathscr{F}_{\mathscr{S}} = \{f(\mathbf{x}^i) | \mathbf{x}^i \in \mathscr{S}\}$. Note that $|\mathscr{S}| = 1$ for sequential, and $|\mathscr{S}| > 1$ for batch sampling algorithms. In each iteration, the newly evaluated points are appended to $\mathscr{D}$, $\mathscr{D} = \mathscr{D} \cup \mathscr{S}$; and their corresponding function values are added to $\mathscr{F}$, $\mathscr{F} = \mathscr{F} \cup \mathscr{F}_{\mathscr{S}}$. The algorithm iterates until a termination criterion (e.g., maximum evaluation budget) is satisfied, and returns the best observed solution, $\mathbf{x}^* \in \arg\min_{\mathbf{x} \in \mathscr{D}} f(\mathbf{x})$.

2.2 Surrogate Modeling

To implement SO algorithms, an efficient surrogate model is needed to approximate the expensive black-box function. Several types of surrogate models have been proposed in the literature, including Gaussian Processes (GP), Multivariate Adaptive Regression Splines (MARS), and Radial Basis Function (RBF) (Anahideh et al. 2022). **Gaussian processes (GPs)** (Rasmussen 2004) are widely used for modeling expensive black-box functions due to their flexible yet powerful nature. Let $f(\mathbf{x})$ be the output corresponding to the newly evaluated input vector $\mathbf{x}$, X represents the matrix of already evaluated points in $\mathscr{D}$, and $\mathbf{f}$ be the corresponding vector of function values. The estimated mean and variance for $\mathbf{x}$ can be calculated as $\hat{\mu}(\mathbf{x}) = K(X,\mathbf{x})\Sigma^{-1}\mathbf{f}$ and $\hat{\sigma}(\mathbf{x}) = K(\mathbf{x},\mathbf{x}) - K(X,\mathbf{x})\Sigma^{-1}K(\mathbf{x},X)$, where Σ represents the covariance matrix and K is a positive semidefinite similarity kernel. The general mathematical functional form for a similarity kernel is given by $K(\mathbf{x},\mathbf{x}') = \theta_0 \cdot \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{x}')^T\Theta^{-1}(\mathbf{x}-\mathbf{x}')\right)$. The kernel function is scaled by the hyperparameter θ_0 and is parameterized by a matrix of hyperparameters Θ. The specific form of the matrix Θ depends on the choice of the kernel and may include parameters such as lengthscale and amplitude associated with the chosen kernel. The hyperparameters, denoted as Θ and θ_0, control the behavior of the similarity kernel and need to be tuned. Common kernel functions are Squared Exponential (SE), Matérn, the Periodic kernel, and Linear. Accurate tuning of the kernel and lengthscale is essential for effective GP modeling.

2.3 Candidate Generation via Discretization

Discretizing the continuous solution space is a crucial step in Black-Box Optimization (BBO) and SO. It addresses the computational challenges of evaluating acquisition functions over the continuous space by converting it into a discrete one. Effective discretization reduces the complexity of the optimization problem by restricting the search space to a finite set of discrete points, facilitating the thorough exploration of the search space and focusing on promising areas. This helps in discovering the global optima efficiently. The choice of a robust discretization method is essential for effective optimization. Among the proposed techniques, dynamic coordinate search (Regis and Shoemaker 2013; Nezami and Anahideh 2022) is the most promising approach. However, even in applying the dynamic coordinate search technique, there are hyperparameters to consider, such as the radius rule that determines the search neighborhood size. Selecting an appropriate radius rule is crucial for achieving a balance between exploration and exploitation, influencing optimization performance and convergence rate.

2.4 Acquisition Functions for Sampling

Acquisition functions play a critical role in guiding the search toward promising candidate points for sampling in SO. They balance exploration and exploitation by selecting points that either improve the model's understanding of unknown regions (exploration) or result in the best objective value (exploitation). Through an effective selection of candidate points, acquisition functions facilitate convergence toward the optimal solution. In the following, we will provide a brief overview of common acquisition functions as documented in the BBO literature.

Expected Improvement (EI) Jones et al. (1998) selects the point with the highest expected improvement over the current best objective value. EI considers an explicit emphasis on both exploration and exploitation through the mean and the variance of the model's prediction. The closed-form equation of EI can be written as, $EI(\mathbf{x}) = E(I(\mathbf{x})) = (\hat{\mu}(\mathbf{x}) - f_n^o)\Phi(\frac{\hat{\mu}(\mathbf{x})-f_n^o}{\hat{\sigma}(\mathbf{x})}) + \hat{\sigma}(\mathbf{x})\phi(\frac{\hat{\mu}(\mathbf{x})-f_n^o}{\hat{\sigma}(\mathbf{x})})$, where $I(\mathbf{x}) = \max(0, [\hat{\mu}(\mathbf{x}) - f_n^o]^+)$, and f_n^o is best observed value after n evaluations. The standard normal density and cumulative distribution are denoted by $\phi(\mathbf{x})$ and $\Phi(\mathbf{x})$, respectively.

Upper Confidence Bound (UCB) Srinivas et al. (2010) balances exploration and exploitation by selecting the point with the highest upper confidence bound. The upper confidence bound is calculated as the sum of the mean prediction ($\hat{\mu}(\mathbf{x})$) and a confidence interval term, which is determined by the variance of the model predictions ($\hat{\sigma}(\mathbf{x})$). At any point $\mathbf{x}$, the UCB equation is given by $\alpha_{UCB}(\mathbf{x},\beta_t) = \hat{\mu}(\mathbf{x}) + \beta_t^{0.5}\hat{\sigma}(\mathbf{x})$, where β_t is a hyperparameter that controls the balance between exploration and exploitation at iteration t.

Weighted Score (Wscore) Regis and Shoemaker (2013) is an acquisition function that balances exploration and exploitation by assigning scores to candidate points based on a linear aggregation over a distance metric and the estimated objective value obtained from the surrogate model. The acquisition function for Wscore at any candidate solution $\mathbf{x} \in \mathscr{C}$ is given by $W^t(\mathbf{x}) = w_d^t \times V_d^t(\mathbf{x}) + w_r^t \times V_r^t(\mathbf{x})$, where w_d^t and w_r^t are predefined weight patterns such that $w_d^t = 1 - w_r^t$, and $V_d^t(\mathbf{x})$ and $V_r^t(\mathbf{x})$ are functions that return normalized scores for the distance metric and the estimated objective value, respectively. The weights w_d^t and w_r^t determine the balance between exploration and exploitation in the acquisition function.

2.5 Thompson Sampling

Thompson Sampling (Thompson 1933) is a probabilistic decision-making strategy commonly used in both SO (Kandasamy et al. 2018) and Multi-Armed Bandit (MAB) (Agrawal and Goyal 2012) problems. In SO, Thompson Sampling constructs a posterior distribution over the function based on the observed data and samples candidate points according to their probability of being the global optimum. Beta distributions are often used as the prior distribution for SO problems in Thompson Sampling. In MAB problems, Thompson Sampling constructs a posterior distribution over the reward distribution of each arm based on the observed rewards and selects the arm with the highest expected reward. The choice of prior distribution depends on the problem and the available prior knowledge. By balancing exploration and exploitation in a probabilistic way, Thompson Sampling can lead to faster convergence to the optimal solution.

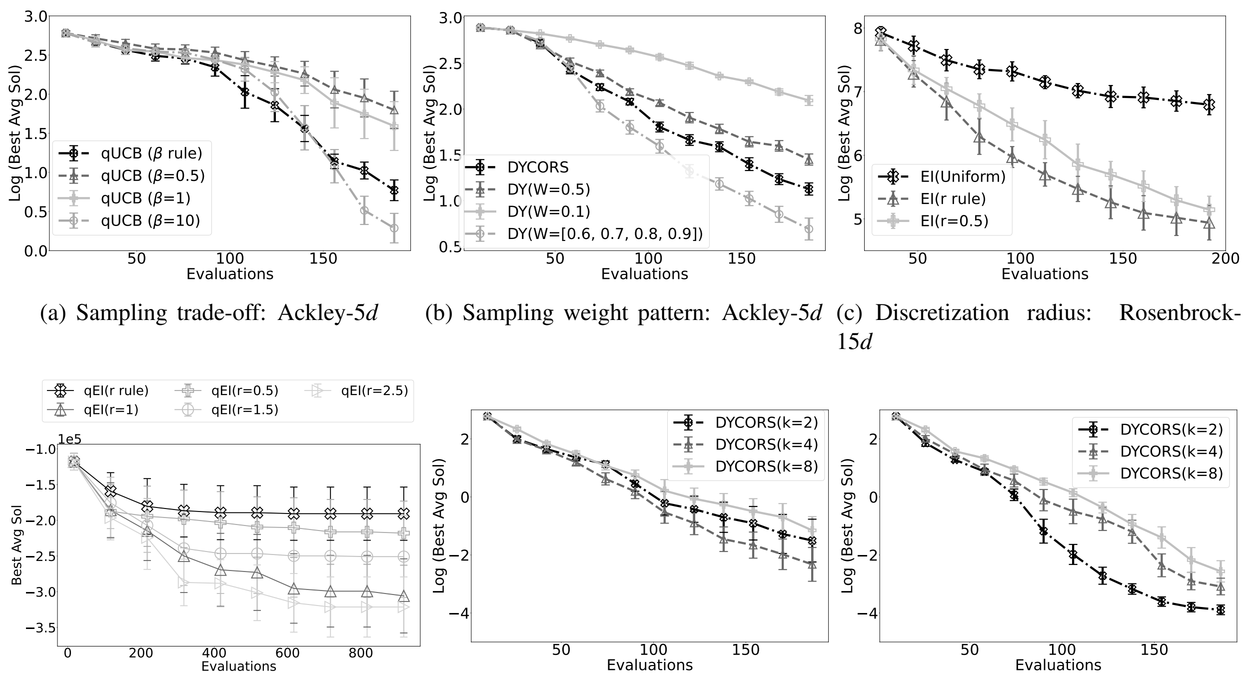
3 IMPACT OF ALGORITHMIC HYPERPARAMETERS ON SURROGATE OPTIMIZATION

As discussed in Section 2, the choice of hyperparameters can significantly affect the performance of SO algorithms. Hyperparameters are tunable parameters that exist in different stages of a given SO algorithm, such as modeling, discretization, and sampling. For instance, SO algorithms that adopt GPs probabilistic models as a surrogate often include hyperparameters such as the kernel function, lengthscale, and the prior probability distribution. The choice of surrogate hyperparameters can influence the convergence rate, the ability to find the global optimum, and the exploration-exploitation trade-off, which in turn affects the overall performance of the algorithm. A poor choice of hyperparameters can result in slow convergence or getting stuck in local optima, leading to suboptimal performance of the algorithm. In this section, we elaborate on this phenomenon through illustrative examples shown in Figure 1. Specifically, we demonstrate the impact of hyperparameters on the Ackley global optimization test problem, utilizing the batch version of Upper Confidence Bound (qUCB), and Dynamic Coordinate Search (DYCORS) (Regis and Shoemaker 2013) in a 5-dimensional setting, shown in Figures 1(a) and 1(b), respectively.

Figure 1(a) shows the impact of the trade-off control hyperparameter β in the qUCB acquisition function, where different initial values for β or a fixed rule to update it in each iteration significantly impact the qUCB convergence rate and the final solution obtained. Similarly, DYCORS, Figure 1(b), which uses the Wscore acquisition to assign scores to candidate points for sampling, is severely impacted by the weight pattern (w_d^t, w_r^t) considered in each iteration. Furthermore, Figures 1(c) and 1(d) reveal the importance of the quality of the generated candidate points, through a discretization schema, for sampling with the EI acquisition function. Figure 1(c) illustrates the impact of the dynamic coordinate-based discretization approach proposed in section 2.3 on the performance of the Expected Improvement (EI) acquisition function compared to the random uniform discretization schema for the 15-dimensional Rosenbrock global optimization test problem. The results demonstrate the disadvantage of using the random uniform discretization schema for EI. Additionally, our experiments on real-world DNA binding test problems, as shown in Figure 1(d), highlight the importance of the radius hyperparameter in the distribution of the generated candidate points resulting from a dynamic discretization strategy. It is worth noting that this phenomenon becomes even more complex when employing SO to tackle real-world problems, particularly in the absence of domain-specific expert knowledge. For instance, the relationship between the structure of DNA and its binding affinity with proteins is highly complex and cannot be easily modeled using a simple mathematical expression.

Therefore, identifying the specific algorithmic criteria required to obtain an optimal solution for this problem is a significant challenge. Hence, accurate tuning of algorithmic hyperparameters is crucial to ensure the effectiveness and efficiency of SO algorithms.

While the impact of individual algorithmic hyperparameters cannot be disregarded, the potential interactions among different hyperparameters in various SO algorithms pose an additional challenge, as shown in Figures 1(e) and 1(f). In these examples, we use the DYCORS algorithm with the Ackley-5d test problem to illustrate the impact of changes in the radius hyperparameter for discretization and the batch size (k) for sampling. Specifically, we compare the results obtained with a fixed radius of 1 in Figure 1(e) to those obtained using the r-rule strategy in Figure 1(f), while varying the batch size. Our experimental results clearly show that the choice of both the batch size and radius hyperparameters has impacted the ability of the algorithm to find optimal solutions, and the impact of these hyperparameters cannot be considered in isolation. It is important to carefully consider the interplay between hyperparameters when designing and implementing SO algorithms. Hence, the complex interactions between hyperparameters can make the optimization process more challenging, necessitating careful experimentation and analysis to identify the best combination of hyperparameter settings. This underscores the need for thorough sensitivity analysis and hyperparameter tuning in SO applications, to ensure reliable results.



(a) Sampling trade-off: Ackley-5d (b) Sampling weight pattern: Ackley-5d (c) Discretization radius: Rosenbrock-15d

(d) Discretization radius: DNA Binding (e) Radius=1: Ackley-5d (f) Radius with r-rule: Ackley-5d

Figure 1: Performance of different SO algorithms under modified hyperparameters values.

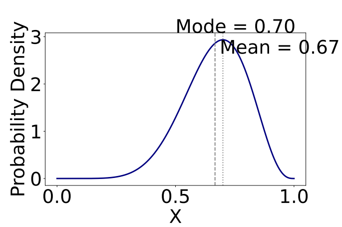
4 HASSO: HYPERPARAMETER ADAPTIVE SEARCH FOR SURROGATE OPTIMIZATION

We present our self-adjusting method for adaptively tuning hyperparameters of SO while optimizing the primary objective. HASSO is versatile and can work with different SO algorithms, regardless of the surrogate model or acquisition/sampling function used. Although we focus on sequential BBO methods (Huang et al. 2006; Shahriari et al. 2015) in this study, our method can also be applied to parallel settings that involve batch evaluations (Krityakierne et al. 2016; Wu and Frazier 2016).

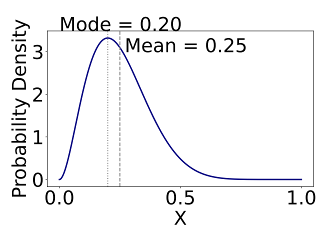
Taking inspiration from the widely-used Thompson Sampling approach for solving the MAB problem (Agrawal and Goyal 2012), HASSO incorporates a hyperparameter tuning step within an SO algorithm. It treats each hyperparameter of a given SO algorithm as a separate arm with an unknown reward distribution. HASSO utilizes the beta distribution which has two shape parameters, α and β, that can be interpreted as the number of successes and failures observed up to the current iteration, respectively. To comply with the purpose of SO, HASSO considers the primary objective improvement of the associated BBO problems as success, and failure is defined as observing no improvement over the current observed objective value. Initially, the prior distribution is set to be a uniform distribution, with $\alpha = \beta = 1$, which means that HASSO has no prior preference over which arm (hyperparameter) to select. Subsequently, the associated beta distribution of each arm will be updated iteratively to influence the probability of hyperparameter selection. In each iteration, $\alpha = \alpha + 1$ when the optimization results in objective improvement (success) or $\beta = \beta + 1$ if no improvement is achieved (failure).

HASSO aims to balance between exploring new hyperparameter configurations and exploiting the best-observed configurations that align with near-optimal regions. By updating the beta distributions based on the actual objective improvements, the method favors exploring multiple promising hyperparameter configurations rather than being restricted to a fixed setting. The accuracy of surrogate models such as GP, can be greatly affected by the choice of hyperparameters. Suboptimal choices can lead to inaccurate models and subpar optimization performance. HASSO addresses this challenge by introducing a stochastic element to mitigate the impact of occasional inaccuracies in the surrogate models due to manual tuning or fixed settings. Instead of solely focusing on surrogate model accuracy which is prone to local optima convergence, HASSO evaluates the overall solution quality of the SO algorithm to assess its effectiveness in achieving the global optimization goal.

Figure 2 demonstrates how the posterior beta distributions of two given hyperparameters in HASSO can change after 20 iterations. Assuming that the first hyperparameter follows a beta distribution with parameters (α_1, β_1), and the second one follows a beta distribution with parameters (α_2, β_2). Moreover, each hyperparameter has been selected 10 times for modification ($\alpha_1 + \beta_1 = \alpha_2 + \beta_2$). Figure 2(a) shows the probability distribution for the first hyperparameter which resulted in 7 successes (objective improvement) and 3 failures (no improvement). Note that we initialize $\alpha_1 = \beta_1 = 1$; therefore, since we update either value by one per iteration, $\alpha_1 = 8$ after 7 successes and $\beta_1 = 4$ after 3 failures. Similarly, Figure 2(b) shows the probability density for the second hyperparameter which resulted in 2 successes and 8 failures. As the fraction of $\frac{\alpha}{\alpha+\beta}$ increases (more success than failure), the probability distribution is more concentrated towards the upper end of the range and results in a higher mean. On the other hand, if the fraction of $\frac{\beta}{\alpha+\beta}$ increases, the distribution is more spread out and concentrated towards the lower end of the range which results in a lower mean. Comparing the two distributions, HASSO is more likely to select the first hyperparameter due to its considerably higher mean value.



(a) $\alpha_1 = 8, \beta_1 = 4$



(b) $\alpha_2 = 3, \beta_2 = 9$

Figure 2: Impact of beta distribution on the selection probability of each hyperparameter.

Algorithm 1 outlines the key steps of our proposed HASSO algorithm, which can be used with any SO approach, given that the acquisition function (e.g., EI) and the surrogate model, along with their associated hyperparameters, are provided. The set of algorithmic hyperparameters and their associated ranges should be predefined in an input set $\mathscr{H}$. Each hyperparameter h_j is initialized with a value v_j, where $a_j \leq v_j \leq b_j, \forall j \in \{1, \ldots, J\}$. For each hyperparameter h_j, HASSO assigns a beta distribution B_j with initialized shape parameters of $\alpha_j = \beta_j = 1$. The set of algorithmic hyperparameters used in each iteration $\mathscr{H}^t$, is first initialized using $\mathscr{H}$ which holds the best set of values obtained so far for each hyperparameter. Then, the algorithm begins by drawing a random value from the associated beta distribution for each hyperparameter to determine its probability of selection. The arm with the maximum value is then selected for an update, denoted by $k = \arg\max_{j \in 1,\ldots,J} u_j$ where u_j is a random sample drawn from the beta distribution for each hyperparameter $j \in \{1,\ldots,J\}$. HASSO can use any updating rule to modify the value of the selected hyperparameter h_k to $v_k^{'}$, and stores the set of updated hyperparameters in $\mathscr{H}^t$.

After selecting the arm with the maximum value, the updated hyperparameter value is utilized in various components of the SO algorithm. The updated hyperparameter is passed to step 14 of Algorithm 1 and subsequently to SO Algorithm 2 in each iteration through $\mathscr{H}^t$. For instance, if hyperparameter h_k is associated with the surrogate model, its value is updated, and the modified model M is utilized to determine the newly selected candidate point $\mathbf{x}^t$ in step 4 of Algorithm 2. Moreover, to facilitate the selection of promising candidate points, the solution space is discretized using the hyperparameter values in $\mathscr{H}^t$. This discretization is denoted as $\mathscr{C}$ and it represents a set of unevaluated points in the solution space. The predefined SO strategy, incorporating the updated model M, the discretized set of points $\mathscr{C}$, and the predefined acquisition function AF, then selects the most promising candidate point $\mathbf{x}^t$ for evaluation in iteration t. In HASSO, improvement (IMP) is defined as the difference between the best observed objective value obtained after evaluating the selected candidate $\mathbf{x}^{*(t)}$ and the best objective value obtained in the previous iteration $\mathbf{x}^{*(t-1)}$. If the updated hyperparameter setting $\mathscr{H}^t$ leads to the evaluation of a more promising candidate point ($IMP > 0$), the algorithm considers it a success. Otherwise, it is considered a failure. In the case of successful sampling, HASSO increases the value of α_k associated with the beta distribution of hyperparameter k which raises the probability of selecting hyperparameter k in future iterations. Additionally, the updated hyperparameter setting $\mathscr{H}^t$ is stored in the $\mathscr{H}$ set, allowing it to be used in subsequent iterations unless it is modified again. On the other hand, when there is no objective improvement, HASSO increases the value of β_k associated with hyperparameter k. By increasing β_k, the probability of selecting hyperparameter k decreases in future iterations. This adjustment helps HASSO focus on exploring hyperparameter values that may yield better results until a termination condition is met.

4.1 The Impact of Adaptive Hyperparameter Search on Optimization Performance

In this section, we provide an illustrative example of our proposed adaptive hyperparameter search approach, HASSO, and its effectiveness using the 2-dimensional Shubert global optimization test problem, as depicted in Figure 3. Shubert has several local and global minima making it prone to local convergence within a limited budget. In Figure 3, HASSO adaptively updates the discretization radius and the lengthscale of the GP model based on the importance of each hyperparameter in optimizing the Shubert function. In contrast, the common practice algorithmic setting, referred to as SO, employs a fixed algorithmic setting for both hyperparameters, primarily focusing on optimizing the Shubert function without employing algorithmic hyperparameter adjustments. The comparison between HASSO and SO reveals that HASSO converges to the near-optimal region faster due to dynamically adjusting the hyperparameter values. Furthermore, HASSO takes into consideration the objective improvement for the Shubert problem when updating the values of hyperparameters in each iteration. It actively explores different hyperparameters based on their contribution to the observed objective value improvement to balance the exploration-exploitation trade-off for hyperparameter tuning. This leads to a strategic movement of the best known solution (BKS) as well as the generated candidate points (Unevaluated Points) toward the desired global optima region.

5 EXPERIMENTS

In this section, we present the results of our empirical analysis evaluating the effectiveness of the proposed adaptive search method, *HASSO*, on a diverse range of global optimization test problems. To evaluate the performance of HASSO, we utilize established SO algorithms. The test problems cover both low and high-

Algorithm 1 Hyperparameter Adaptive Search for Surrogate Optimization

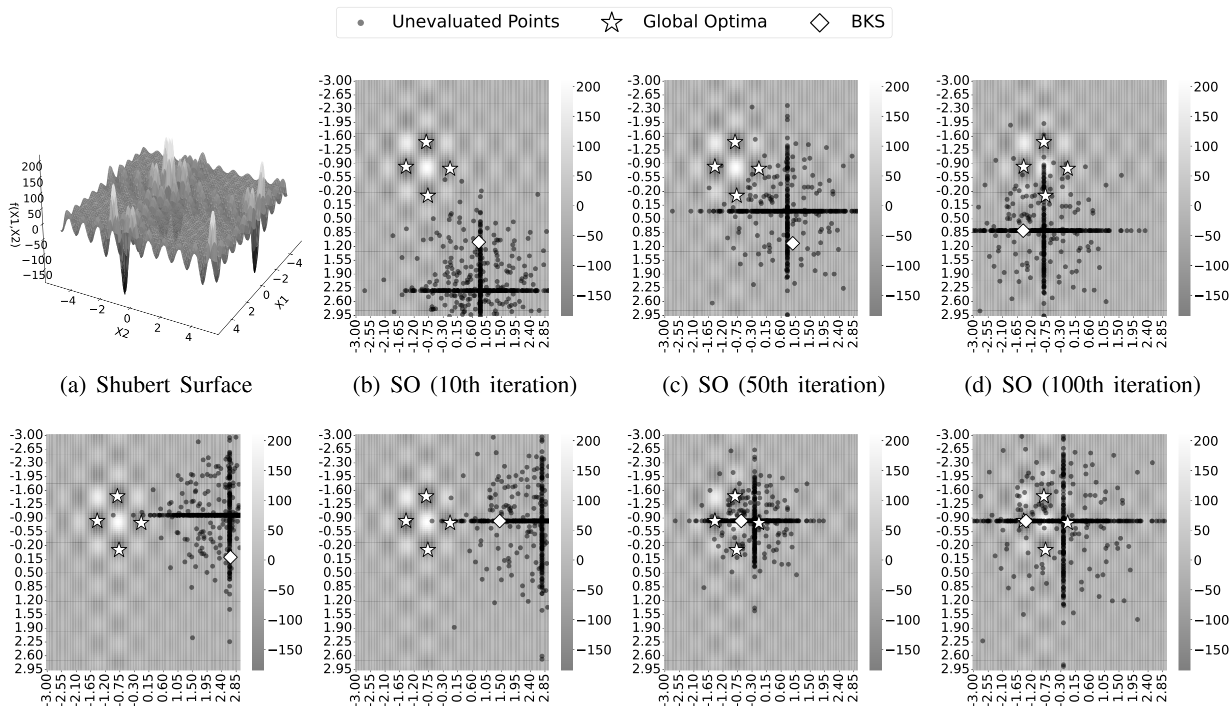
$\mathscr{D} = \{\mathbf{x}^1, \ldots, \mathbf{x}^n\}$, $\mathscr{F} = \{f(\mathbf{x}^i) | \mathbf{x}^i \in \mathscr{D}\}$, $\mathbf{x}^{*(0)} = \arg\min_{\mathbf{x} \in \mathscr{D}} f(\mathbf{x})$, $t = 1$, T=Total Budget,
$AF =$ Acquisition function
$\mathscr{H} = \{h_1 = \nu_1, h_2 = \nu_2, \ldots, h_J = \nu_J | a_j \leq \nu_j \leq b_j, \forall j \in \{1, \ldots, J\}\}$
$B_j \sim Beta(\alpha_j = 1, \beta_j = 1) \forall j \in \{1, \ldots, J\}$.
while $t \leq T$ **do**
  $\mathscr{H}^t \leftarrow \mathscr{H}$
  Choose one arm:
  for $j \in \{1, \ldots, J\}$: **do** u_j= Draw a random sample from $B_j \sim Beta(\alpha_j, \beta_j)$
  end for
  $k = \arg\max_{j \in \{1,\ldots,J\}} u_j$
  Update the Hyperparameter:
  $\nu_k^{'}$= Update h_k s.t $a_k \leq \nu_k \leq b_k$
  $\mathscr{H}^t = \{h_1 = \nu_1, h_2 = \nu_2, \ldots, h_k = \nu_k^{'}, \ldots, h_J = \nu_J\}$
  $\mathbf{x}^t$= Srg-opt($\mathscr{D}, \mathscr{F}, \mathscr{H}^t, AF$)
  Evaluation: $\mathscr{F}_{\mathbf{x}^t} = f(\mathbf{x}^t)$
  $\mathscr{D} = \mathscr{D} \cup \mathbf{x}^t$; $\mathscr{F} = \mathscr{F} \cup \mathscr{F}_{\mathbf{x}^t}$
  $\mathbf{x}^{*(t)} = \arg\min_{\mathbf{x} \in \mathscr{D}} f(\mathbf{x})$
  $IMP = \mathbf{x}^{*(t-1)} - \mathbf{x}^{*(t)}$
  if $IMP > 0$ **then**: $\mathscr{H} \leftarrow \mathscr{H}^t$, $\alpha_k = \alpha_k + 1$
  else: $\beta_k = \beta_k + 1$
  end if
  $\mathbf{x}^{*(t)} = \arg\min_{\mathbf{x} \in \mathscr{D}} f(\mathbf{x})$
  $t = t + 1$
end while
Return $\mathbf{x}^* = \mathbf{x}^{*(T)}$

Algorithm 2 Srg-opt

Input: $\mathscr{D}, \mathscr{F}, \mathscr{H}, AF$
$M =$ Fit a surrogate model $\hat{f}$ on $(\mathscr{D}, \mathscr{F})$ using related $\mathscr{H}$ hyperparameter values
$\mathscr{C}$= Discretize the solution space using the hyperparameters in $\mathscr{H}$
Determine new candidate point, $\mathbf{x}$, using M,$\mathscr{C}$, and AF with hyperparameters in $\mathscr{H}$
Return $\mathbf{x}$

dimensional settings, and we examine the performance of HASSO across different problem dimensions and search budgets. We also analyze the behavior of HASSO by investigating the impact of different initializations on its performance.

Baselines. The experiment section includes several baseline scenarios that were considered in our study. In the first scenario, denoted as *SO*, the surrogate optimization algorithm employs a fixed initialization for all hyperparameters. The second scenario, *SO-2*, is based on the r-rule, as proposed by Regis and Shoemaker (2013), to update the radius for candidate generation while all other hyperparameters are fixed. The third scenario, *SO-grid*, involves designing a grid based on the range of each hyperparameter and searching for the best setting under each. Finally, the fourth scenario, *SO-rand*, performs random updates on all hyperparameters in each iteration. We examine two alternative strategies for the HASSO algorithm, achieved by modifying the hyperparameter updating strategy as described in Algorithm 1. The first strategy, denoted as *HASSO-decay*, employs a decay function to update the hyperparameters based on the trade-off between exploration and exploitation. Specifically, the *HASSO-decay* strategy adjusts the hyperparameter



(a) Shubert Surface (b) SO (10th iteration) (c) SO (50th iteration) (d) SO (100th iteration)

(e) Initial State (f) HASSO (10th iteration) (g) HASSO (50th iteration) (h) HASSO (100th iteration)

Figure 3: Impact of adaptive hyperparameter Shubert ($2d$).

values differently for positive and non-positive improvement, aiming to promote more exploitation (e.g., by reducing the lengthscale value in the GP kernel) when positive improvement is observed, and more exploration (e.g., by increasing the radius) when no improvement is achieved. As the optimization process proceeds, the decay function reduces the magnitude of the updates to the hyperparameters. In contrast, the second strategy, denoted as *HASSO-rand*, uses only random updates for each hyperparameter, within its predefined range. The unrestricted updates considered by *HASSO-rand* allow the algorithm to explore a wider range of hyperparameter values, potentially enhancing the overall search performance. The summary of all scenarios is provided in Table 1.

Table 1: Experiment baselines comparison.

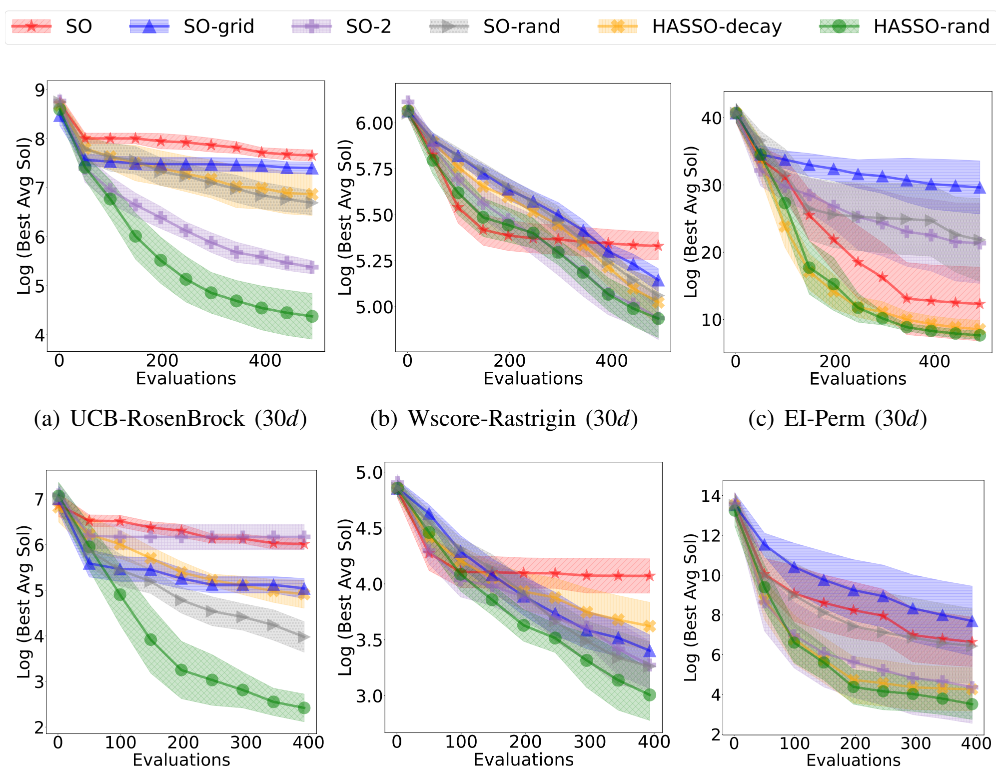
Algorithm	Srg hyperparameter	Dis hyperparameter	Update
SO	fixed	fixed	None
SO-2	fixed	r-rule	factor 2
SO-grid	grid-search to find best	grid-search to find best	use the best
SO-rand	random change	random change	random
HASSO-decay	adaptive decay change	adaptive decay change	decay function
HASSO-rand	adaptive random change	adaptive random change	random

Experiment Set-up: In this study, we investigate the performance of the considered baselines in solving three global optimization test problems that possess diverse shapes and topological characteristics in both medium and high-dimensional settings. To evaluate each test problem, we employ the Latin Hypercube Design method with the maximin criterion to generate distinct sets of initial points, each consisting of $2*(d+1)$ evaluations. We repeat the experiments 30 times, using each set of initial points. In medium-dimensional experiments, the budget is set to 400 and the discretized solution space size is 1000. In medium-dimensional problems, the budget is set to 500 and the discretized solution space size is 4000.

5.1 Empirical Results

Figure 4 presents the results of our empirical analysis, which compares the performance of our proposed method, HASSO, with other baselines. We evaluated our method on three global optimization problems that exhibit different topological characteristics across both mid-dimensional and high-dimensional settings. Rosenbrock is a unimodal valley-shaped function of the form $\sum_{i=1}^{d-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$, Rastrigin is a multimodal function of the form $10d+\sum_{i=1}^{d}[x_i^2-10cos(2\pi x_i)]$, and Perm is a bowl-shaped function of the form $\sum_{i=1}^{d}(\sum_{j=1}^{d}(j+\beta)(x_j^i-\frac{1}{j^i}))^2$. To solve these problems, we employed commonly used SO algorithms, namely UCB, Wscore, and EI. We also considered two hyperparameters, namely the lengthscale for the Matern kernel in the GP model, and the radius used for dynamic candidate generation.

Based on the experimental results presented in Figure 4, we observed that using fixed hyperparameter values in SO algorithms can lead to suboptimal results compared to using random updates (SO-rand) for most test problems. Incorporating random updates in each iteration can lead to faster convergence and better results. In contrast, defining a fixed discrete grid for each hyperparameter and searching for the best setting hierarchically (SO-grid) might be worse than the fixed setting (SO) for smooth bowl-shaped problems with a unique global optimum, such as Perm. The SO-2 strategy, which applies the r-rule update in a fixed setting, was competitive with the SO, SO-grid, and SO-rand baselines. Furthermore, we found that the HASSO-decay performance is competitive with HASSO-rand for the smooth Perm test problem. However, the decay in the updates puts HASSO-decay at a significant disadvantage for test problems that require more exploration, such as Rosenbrock and Rastrigin, especially with increasing dimensionality. We found that HASSO-rand performs better than other baselines in all the test problems we considered. Specifically, HASSO is highly effective in global optimization problems with diverse topological characteristics.



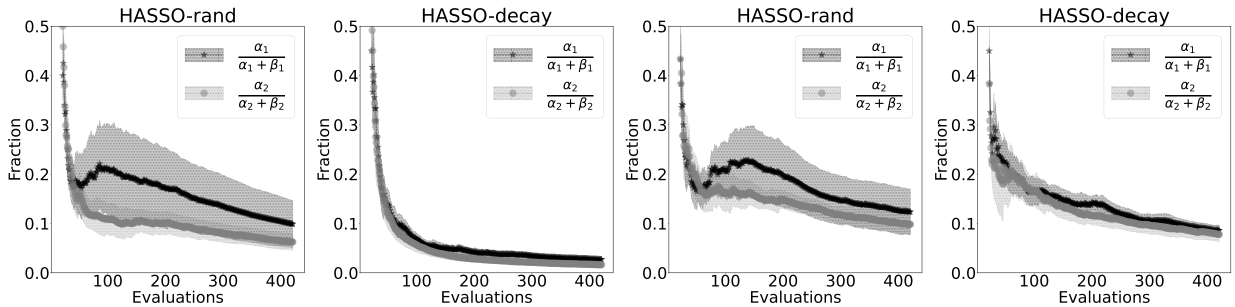
(a) UCB-RosenBrock (30d) (b) Wscore-Rastrigin (30d) (c) EI-Perm (30d)

(d) UCB-RosenBrock (10d) (e) Wscore-Rastrigin (10d) (f) EI-Perm (10d)

Figure 4: Performance comparison of HASSO and other baselines for different BBO algorithms and global optimization test problems.

5.2 HASSO: Analysis on the Updating Rule

In Figure 5, we present the average $\frac{\alpha}{\alpha+\beta}$ fractions for the two hyperparameters in the test problems shown in Figure 4. The findings affirms that HASSO-rand outperforms HASSO-decay in terms of success rate (objective improvements) in both problems. This distinction is particularly pronounced for Rosenbrock. Furthermore, for HASSO-rand, the first hyperparameter (GP's lengthscale) demonstrates considerably more success compared to the second hyperparameter (radius). In contrast, HASSO-decay has a similar number of successes and failures for both hyperparameters.



(a) UCB-RosenBrock(10*d*) (b) UCB-RosenBrock(10*d*) (c) EI-Perm(10*d*) (d) EI-Perm(10*d*)

Figure 5: Impact of the update rule by comparing HASSO-rand with HASSO-decay.

In Figure 6, the boxplots present a comparison of the computational time required for a full run (i.e., one repetition) of our considered baselines for high-dimensional (30*d*) global optimization test problems. The results show that incorporating HASSO into any regular SO algorithm results in only a negligible increase in computational time compared to the SO algorithm without HASSO (the SO baseline). Additionally, the computational time required for HASSO is insignificantly different from that of applying any random update (SO-rand) or rule (SO-2). However, it is worth noting that using a hierarchical grid search to identify the best hyperparameters can significantly reduce search efficiency.

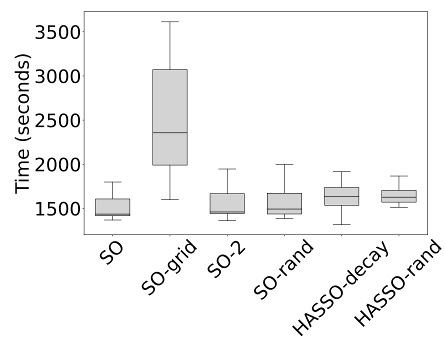


Figure 6: Computational time to complete one full run of 30*d* test problems.

6 CONCLUSION

This study sheds light on the crucial role played by hyperparameters in the performance of expensive black-box optimization algorithms. Manual tuning of these hyperparameters can be challenging and time-consuming, hindering the adoption of SO approaches across different problem domains. To address this issue, we proposed HASSO, an adaptive hyperparameter optimization framework that automatically adjusts hyperparameters based on the algorithm's outcome in each iteration of optimization. Our experimental results demonstrate that HASSO significantly improves the performance of various optimization algorithms while reducing the need for manual tuning. Future research directions could explore the extension of HASSO to parallel optimization settings and investigate its performance in problems involving more than two hyperparameters. Additionally, testing the proposed approach on real-world optimization problems would provide valuable insights into its practical applicability. Overall, our study highlights the potential of adaptive hyperparameter optimization approaches like HASSO in enhancing the efficiency and effectiveness of black-box optimization algorithms.

AUTHOR BIOGRAPHIES

NAZANIN NEZAMI is a Ph.D. Student in the Mechanical and Industrial Engineering Department at the University of Illinois Chicago (UIC). She obtained her M.S. degree in Industrial and Systems Engineering from the University of Minnesota Twin Cities prior to joining UIC. Her main research interests are Black-Box Optimization, Machine Learning (ML), and Fairness in ML. Her email address is nnezam2@uic.edu.

HADIS ANAHIDEH is an Assistant Professor in the Mechanical and Industrial Engineering Department at the University of Illinois Chicago. She received her Ph.D. degree in Industrial Engineering from the University of Texas at Arlington. Her research objectives center around Black-box Optimization, Sequential Optimization, Active Learning, Statistical Learning, Explainable AI, and Algorithmic Fairness. Her email address is hadis@uic.edu and her homepage can be found at https://mie.uic.edu/profiles/anahideh-hadis/.

REFERENCES

Agrawal, S., and N. Goyal. 2012. "Analysis of Thompson Sampling for the Multi-armed Bandit Problem". In *Conference on learning theory*, 39–1. Journal of Machine Learning Research Workshop and Conference Proceedings.

Anahideh, H., J. Rosenberger, and V. Chen. 2022. "High-dimensional Black-box Optimization Under Uncertainty". *Computers & Operations Research* 137:105444.

Benjamins, C., A. Jankovic, E. Raponi, K. van der Blom, M. Lindauer, and C. Doerr. 2022. "Towards Automated Design of Bayesian Optimization via Exploratory Landscape Analysis". In *6th Workshop on Meta-Learning at Conference on Neural Information Processing Systems 2022*. MIT Press.

Chen, W., K. Chiu, and M. Fuge. 2019. "Aerodynamic Design Optimization and Shape Exploration Using Generative Adversarial Networks". In *American Institute of Aeronautics and Astronautics Scitech 2019 Forum*, 2351. San Diego, California: American Institute of Aeronautics and Astronautics.

Feurer, M., and F. Hutter. 2019. "Hyperparameter Optimization". In *Automated Machine Learning*, edited by L. K. Frank Hutter and J. Vanschoren, 3–33. Cham: Springer, Cham.

Huang, D., T. T. Allen, W. I. Notz, and N. Zeng. 2006. "Global optimization of stochastic black-box systems via sequential kriging meta-models". *Journal of global optimization* 34:441–466.

Jafar, S. H. 2022. "Financial Applications of Gaussian Processes and Bayesian Optimization". In *Bayesian Reasoning and Gaussian Processes for Machine Learning Applications*, 111–122. Chapman and Hall/CRC.

Jones, D. R., M. Schonlau, and W. J. Welch. 1998. "Efficient Global Optimization of Expensive Black-box Functions". *Journal of Global Optimization* 13(4):455–492.

Kandasamy, K., A. Krishnamurthy, J. Schneider, and B. Póczos. 2018. "Parallelised Bayesian Optimisation via Thompson sampling". In *International Conference on Artificial Intelligence and Statistics*, 133–142. Proceedings of Machine Learning Research.

Krityakierne, T., T. Akhtar, and C. A. Shoemaker. 2016. "SOP: Parallel Surrogate Global Optimization with Pareto Center Selection for Computationally Expensive Single Objective Problems". *Journal of Global Optimization* 66(3):417–437.

Malkomes, G., and R. Garnett. 2018. "Automating Bayesian Optimization with Bayesian Optimization". In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 5988–5997. Red Hook, NY, USA: Curran Associates Inc.

Nezami, N., and H. Anahideh. 2022. "An Empirical Review of Model-Based Adaptive Sampling for Global Optimization of Expensive Black-Box Functions". In *2022 Winter Simulation Conference*, edited by B. Feng, G. Pedrielli, Y. Peng, S. Shashaani, E. Song, C. Corlu, L. Lee, and P. Lendermann, 3182–3193. Institute of Electrical and Electronics Engineers.

Pyzer-Knapp, E. O. 2018. "Bayesian Optimization for Accelerated Drug Discovery". *IBM Journal of Research and Development* 62(6):2–1.

Rasmussen, C. E. 2004. *Gaussian Processes In Machine Learning*, 63–71. Berlin, Heidelberg: Springer Berlin Heidelberg.

Regis, R. G., and C. A. Shoemaker. 2013. "Combining Radial Basis Function Surrogates and Dynamic Coordinate Search in High-dimensional Expensive Black-box Optimization". *Engineering Optimization* 45(5):529–555.

Shahriari, B., K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas. 2015. "Taking the Human Out of the Loop: A Review of Bayesian Optimization". *Proceedings of the IEEE* 104(1):148–175.

Srinivas, N., A. Krause, S. Kakade, and M. Seeger. 2010. "Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design". In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, 1015–1022. Madison, WI, USA: Omnipress.

Thompson, W. R. 1933. "On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples". *Biometrika* 25(3-4):285–294.

Wu, J., and P. Frazier. 2016. "The Parallel Knowledge Gradient Method for Batch Bayesian Optimization". *Advances In Neural Information Processing Systems* 29.