Received 29 March 2023, accepted 22 April 2023, date of publication 27 April 2023, date of current version 5 May 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3271134

RESEARCH ARTICLE

Adaptive Bayesian Optimization for Fast Exploration Under Safety Constraints

GUK HAN, JONGOH JEONG, (Graduate Student Member, IEEE), AND JONG-HWAN KIM, (Life Fellow, IEEE)
School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon 34141, Republic of Korea

Corresponding author: Jong-Hwan Kim (johkim@rit.kaist.ac.kr)

This work was supported by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by the Korean Government (Ministry of Science and ICT (MSIT); 2020-0-00842, ‘‘Development of Cloud Robot Intelligence for Continual Adaptation to User Reactions in Real Service Environments’’).

ABSTRACT The industrial field faces the problem of process optimization by finding the factors affecting the yield of the process and controlling them appropriately. However, due to limited resources such as time and money, optimization is performed using a low evaluation budget. In addition, for process stability, the lower limit of the yield is set so that the yield must be maintained above this limit during optimization. Bayesian Optimization (BO) can be an effective solution in acquiring optimal samples that satisfy a safety constraint given a low evaluation budget. However, many existing BO algorithms have some limitations such as significant performance degradation due to model misspecification, and high computational load.Thus, we propose a practical safe BO algorithm, A-SafeBO, that effectively reduces performance degradation due to model misspecification using only a limited evaluation budget. Additionally, our algorithm performs computations for a large number of observations and high-dimensional input spaces by using Ensemble Gaussian Processes and Safe Particle Swarm Optimization. Here, we also propose a new acquisition function that leads to a wider exploration even under the constraint of safety. This will help deviate from the local optimum and achieve a better recommendation. Our algorithm empirically guarantees convergence and performance through evaluations on several synthetic benchmarks and a real-world optimization problem.

INDEX TERMS Bayesian optimization, Gaussian processes, safe exploration, hyperparameter scaling, industrial application, low evaluation budget environment.

I. INTRODUCTION

Over the past few decades, the accumulation of high-quality data and the development of computing power have continually been developed. More recently, machine learning has been used to extract meaningful information from a large amount of high-quality data that is difficult for humans to directly perform, leading to countless advances in various fields.

A typical example is the industrial sector. Industrial sites aim to produce high-quality products with high yield through the production process. To this end, accumulated production data can now be analyzed using machine learning to understand the relationship between the control parameters affecting the production process and the resulting product quality and yield. It can then adjust the control parameters appropriately to derive high productivity.

However, this approach is not always available because the accumulated production data sometimes loses its meaning. The yield of the product is affected not only by the control parameters of the machine, but also by external environmental factors such as temperature and humidity. Although many efforts have been made to control these external environmental factors in the production line, sometimes these factors affect the production line and change the correlation between the previously identified control parameters of the machine and changes in production yield; the previously collected data then loses meaning. In addition, if the type of product to be produced on the production line is suddenly changed, the previously collected data also loses meaning.

The associate editor coordinating the review of this manuscript and approving it for publication was Jolanta Mizera-Pietraszko.

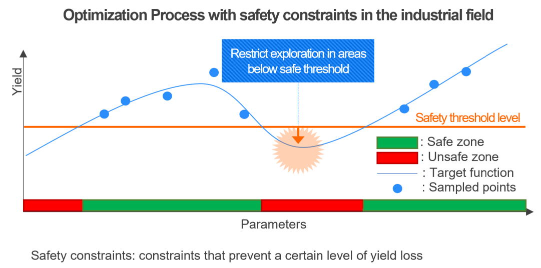
Optimization Process with safety constraints in the industrial field
Restrict exploration in areas below safe threshold
Yield
Safety threshold level
🟩 : Safe zone
🟥 : Unsafe zone
: Target function
● : Sampled points
Parameters
Safety constraints: constraints that prevent a certain level of yield loss

FIGURE 1. In the optimization process during the production process in the industrial field, there are cases in which exploration is restricted to lower the yield below a specific value in order to prevent large financial losses.

In this situation, the accumulated production data is very small; therefore, data collection should be prioritized to analyze the relationship between the control parameters of the production machine and the production yield. However, it is difficult to have a data collection process that requires production to stop due to the nature of the industrial environment in which stopping production itself causes a huge financial loss. In addition, since the process of manufacturing a product for data collection itself consumes many resources (materials, time, manpower, etc.), it is necessary to effectively analyze this relationship using a relatively small amount of data.

Moreover, a single wrong choice of control parameters in the industrial sector can lead to damage to production machinery and result in significant financial losses. Therefore, data collection that could cause such harm must be avoided. To address this issue, a safe exploration technique is necessary that can gather the most information while accurately predicting the risk associated with each evaluation based on the collected data. As depicted in Figure 1, this technique should enable evaluations that are not risky.

Although model-based deep neural network (DNN) approaches have recently shown great success in many fields, the use of DNNs is unsuitable in settings where limited data and safety is critical. This is because DNNs consume a lot of time and data to train the model. DNNs also require careful tuning of model hyperparameters (e.g. batch size, learning rate, number of nodes in hidden layer, etc); otherwise, random evaluation may incur high physical and financial opportunity costs [1], [2].

Bayesian optimization (BO) is an effective method to solve this problem. BO is a well-principled, sample-based approach that acquires data sequentially and optimizes black box objectives. It operates by building a surrogate model for optimizing the objective by sampling the point that optimizes the acquisition function [3], [4]. This approach has recently gained popularity for its effectiveness in tuning the hyperparameters of various machine learning models, especially DNNs [1], [2]. It has also been employed in practical safety-critical scenarios such as automated manufacturing systems [5], [6] and medical product development [7], [8]. In these applications, the goal is to sequentially draw an optimal sample, satisfying some given safety constraints, from the design space representing the relationship between the control parameters and the reward.

While BO has long stood as a standard optimization method in multi-armed bandit problems [9], it inherently bears limitations in the computation time and number of samples required to reach the global optimum in a high-dimensional search space. In the case of BO, which creates a surrogate model based on $\mathcal{GP}$ [9], [10], the use of appropriate kernel hyperparameters greatly affects performance. However, the process of selecting appropriate kernel hyperparameters usually consumes a significant amount of resources (e.g. training data samples, time). This process does not fit the concept of BO, which is used to efficiently perform optimization using a relatively small number of samples. In addition, when such a large number of samples are randomly collected for this process, the possibility of damaging the environment increases in situations where safety constraints are important. Another concern is the necessary sample size growing in proportion to the search space dimension for global convergence, *i.e.*, more observations are required to arrive at the global optimum in a high dimensional space [11]. In addition, the BO approach is characterized by exponentially increasing computational complexity for the number of samples of n [12]. These characteristics make BO unsuitable for use in industrial sites where it is necessary to obtain optimal sample points on a low evaluation budget and reduce the risk of sampling candidates with unsafe yields.

In this paper, we propose a practical safe BO algorithm called A-SafeBO that overcomes these limitations and achieves competitive performance in safety-critical and data-sparse environments.

Our contributions are summarized as follows:

- We introduce a safe BO algorithm that is practically useful in low evaluation budget and safety-critical environments by reducing computations using Ensemble Gaussian Processes and Safe Particle Swarm Optimization.
- Our algorithm proposes a new acquisition function that induces safer and wider exploration than existing methods in situations where exploration is limited due to safety constraints. Through this, it is possible to give a recommendation closer to the global optimum.
- Our algorithm saves resources used for hyperparameter tuning by effectively performing optimization by scheduling the hyperparameter scaling process within a limited number of samples using the scaling function without a tuning process, even if information on kernel hyperparameters is not given in advance.
- We evaluated our algorithm on several benchmarks and a real-world optimization problem and empirically show that it achieves competitive results in the proportion of safe samples, computation time, and estimated reward compared with the baseline methods [13], [14].

II. PRELIMINARIES

In this section, we describe the underlying principles and assumptions necessary for BO-based approaches.

A. BAYESIAN OPTIMIZATION

BO has gained popularity as an optimization technique for solving expensive objective functions in practical applications [10]. This optimization method is effective in arriving at the global optimum of an expensive objective $f:\mathcal{X}\mapsto\mathbb{R}$ for a compact domain $\mathcal{X}\subset\mathbb{R}^d$. This framework is used to iteratively update the statistical surrogate model and select the next informative sample location by optimizing the acquisition function. The maximizer $\mathbf{x}^*$ on the black box function f is often determined in a one-step optimal manner by balancing the exploration-exploitation trade-off in a greedy fashion. Formally, the goal is to optimize the following problem:

$$\mathbf{x}^* = \underset{x\in\mathcal{X}}{\operatorname{argmax}}\, f(\mathbf{x}), \tag{1}$$

where $\mathcal{X}$ and $f(\cdot)$ denote a d-dimensional design space and the objective function to be evaluated, respectively. Key to global convergence in BO is to appropriately adjust the degree of exploration and exploitation trade-off [15].

B. REGULARITY ASSUMPTIONS

The surrogate function in the BO setting is modeled with a stochastic process on a probability space defining a prior distribution over functions [10]. A typical choice for the nonparametric regression function f is a $\mathcal{GP}$ [16] which is a smooth yet flexible model fully parameterized by a prior mean function and a covariance (kernel) function, *i.e.*, $f \sim \mathcal{GP}(\mu(\mathbf{x}), k_l(\mathbf{x}, \mathbf{x}'))$. With $\mathcal{GP}$, we can assume that $\mathcal{X}$ is provided with a positive definite kernel function, and the objective f has a bounded norm $\mathbf{B}$ in the associated Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}_{k_l}(\mathcal{X})$, which is completely determined by the corresponding kernel function $k_l(\cdot,\cdot) : \mathcal{X}\times\mathcal{X}\mapsto\mathbb{R}$ determining the roughness and size of the function space [17]. The induced bounded RKHS norm $||f||_{k_l}=\sqrt{\langle f,f\rangle}\le\mathbf{B}$ is a measure of the function complexity with respect to the kernel, assuming we have well-behaved functions of the form $f(\mathbf{x})=\sum_{i\ge0}\left(\alpha_i \times k_l(\mathbf{x}, \mathbf{x}_i)\right)$ for points $\mathbf{x}_i\in\mathbb{R}^d$ and weight decay coefficients $\alpha_i\in\mathbb{R}$. These assumptions allow us to estimate f as a sample from $\mathcal{GP}$.

At each iteration $t = \{1, \cdots, T\}$, the $\mathcal{GP}$ prior is updated from the cumulative set of the past training samples $D_t = \{(\mathbf{x_i}, y_i)\}_{i=1}^t$ using the Bayes' rule. For a design vector $\mathbf{x}$ at iteration t, the posterior mean μ_T and variance σ_T are obtained in closed-form as follows:

$$\begin{aligned}\mu_T(\mathbf{x}) &= \mathbf{k}_T(\mathbf{x})^\intercal(\mathbf{K}_T + \mathbf{I}_T\rho^2)^{-1}y_T,\\ k_T(\mathbf{x}, \mathbf{x}') &= k_l(\mathbf{x}, \mathbf{x}') - \mathbf{k}_T(x)^\intercal(\mathbf{K}_T + \mathbf{I}_T\rho^2)^{-1}\mathbf{k}_T(\mathbf{x}'),\\ \sigma_T^2(\mathbf{x}) &= k_T(\mathbf{x}, \mathbf{x}),\end{aligned} \tag{2}$$

where $\mathbf{K}_T\in\mathbb{R}^{T\times T}$ is the covariance matrix with i-th and j-th entries $k_l(\mathbf{x}_i, \mathbf{x}_j)$ for $x_i, x_j\in\mathcal{X}_T$ and $i, j\in\{1, \ldots, T\}$, and the vector $\mathbf{k}_T(\mathbf{x})=[k_l(\mathbf{x}, \mathbf{x}_1), \ldots, k_l(\mathbf{x}, \mathbf{x}_T)]^\intercal$ holds the covariances between the input $\mathbf{x}$ and the observed samples in D_T. We denote the identity matrix as $\mathbf{I}_T\in\mathbb{R}^{T\times T}$. Without loss of generality, we assume $\mu(\mathbf{x})=0$ and our observations are perturbed by ρ-sub-Gaussian noise.

We also assume that f is L-Lipschitz continuous with respect to some metric d on $\mathcal{X}$. The design choice x and the optimal point x' are to satisfy (3) by using a commonly used isotropic kernel on $\mathcal{X}$, *e.g.*, the Gaussian kernel [9], [18].

$$|f(x) - f(x')| \le L\cdot d(x, x') \quad \forall x, \tag{3}$$

where $d(\cdot,\cdot)$ is some distance metric with which the objective f is L-Lipschitz continuous. The estimated function value at any x is assumed not to change at an arbitrary rate, but bounded by the L-Lipschitz constant.

C. ACQUISITION FUNCTION

The next evaluation point $\mathbf{x}_{t+1}$ is then selected by maximizing the utility function U_t such that $\mathbf{x}_{t+1}=\operatorname{argmax}_{x\in\mathcal{X}}U_t(\mathbf{x})$ assuming that optimizing U_t is a maximization problem. We avoid expensive evaluation of the objective f by maximizing this auxiliary function, thus obtaining the posterior quantities of the $\mathcal{GP}$ at a cheaper cost [19]. Under the smoothness assumptions via a $\mathcal{GP}$, the acquisition function (*e.g.*, Upper Confidence Bound (UCB) [20], Probability of Improvement (PI) [21], Expected Improvement (EI) [22], and GP-UCB [9]) serves to balance the exploration-exploitation trade-off. For instance, GP-UCB balances the trade-off with β_t in greedily maximizing $U_t = \mu_{t-1}(\mathbf{x}) + \beta_{t-1}^{1/2}\sigma_{t-1}(\mathbf{x})$

D. KERNEL HYPERPARAMETERS

The performance of BO based on $\mathcal{GP}$ is greatly affected by the kernel hyperparameters of $\mathcal{GP}$. These kernel hyperparameters are determined by the characteristics of the data used to build the $\mathcal{GP}$, and it is practically impossible to know these hyperparameters before collecting the data. Therefore, before use, it must be set appropriately according to the characteristics of the data to be used. This process is often called hyperparameter tuning; however, this process consumes a lot of resources. These problems have made the practical use of BO difficult. Therefore, studies have been conducted to eliminate the hyperparamter tuning process and to adjust the hyperparameters online [23], [24], [25], [26]. In these studies, length scale(l), norm bound($\mathbf{B}$), and beta(β) are cited as common parameters that have a great influence on performance. To model the objective function f using $\mathcal{GP}$, the regularity assumption that the function space of f must be bounded by the RHKS norm bound $\mathbf{B}$ must be satisfied. Since this is a number that limits the complexity of the function by limiting the function space of the objective function f as in $||f||_{k_l}=\sqrt{\langle f,f\rangle}\le\mathbf{B}$, norm bound $\mathbf{B}$ has an important meaning. Therefore, the greater the value of $\mathbf{B}$, the higher the complexity of f.

The kernel mainly used measures its similarity based on the distance between two inputs.The length scale l is a scaling factor for measuring similarity by scaling the scale of each dimension of the input domain when it is varied as (4). In this case, the kernel calculates that it has a large similarity when the distance is close; therefore, having a large similarity when the length scale l is large means that the similarity between

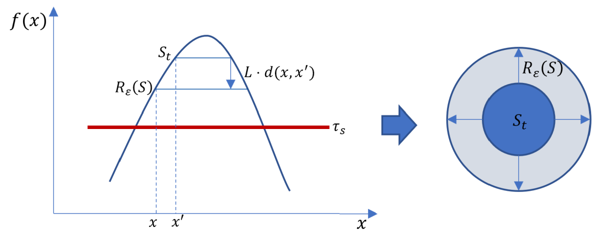

FIGURE 2. Estimate safe regions ($R_\epsilon(S)$) based on the current safe set(S) under the L-Lipschitz continuity assumption.

two points is high even if the distance between them is far. This means that the complexity of this function is low. Conversely, when the length scale l is small, the distance between points with high similarity is small, indicating that the complexity of the function is high.

$$k_l(\mathbf{x}, \mathbf{x}') = k\left(\frac{[\mathbf{x}]_1 - [\mathbf{x}']_1}{[l]_1}, \cdots, \frac{[\mathbf{x}]_d - [\mathbf{x}']_d}{[l]_d}\right) \tag{4}$$

Finally, β is mainly used to define the confidence bound $CB(\cdot)$, the boundary where the observation exists with high probability using the mean function and standard deviation function, which is defined as (5).

$$\begin{aligned}\beta_t^{1/2} &= \mathbf{B}_t + 4\rho\sqrt{I(\mathbf{y}_t; f) + 1 + \ln(1/\delta)} \\ CB_t(\mathbf{x}) &= \mu_t(\mathbf{x}) \pm \beta_t^{1/2}\sigma_t(\mathbf{x}),\end{aligned} \tag{5}$$

where ρ-sub-Gaussian noise is added to the measurements and $I(\mathbf{y}_t; f)$ is the mutual information between the $\mathcal{GP}$ prior on f and the t-th observation $\mathbf{y}_t$. The confidence bound $CB(\mathbf{x})$ is an interval in which samples, $\mathbf{x}$, can exist with a probability of $1-\delta$ (see Appendix A, Lemma 1).

E. SAFE OPTIMIZATION GOAL

Under the Lipschitz continuity assumption (3), we can define a one-step ε-reachability operator [13] based on the safe set S as follows.

$$R_\epsilon(S) := S \cup \left\{\mathbf{x} \in D | \exists \mathbf{x}' \in S, f(\mathbf{x}') - \epsilon - L \cdot d(\mathbf{x}, \mathbf{x}') \geq \tau_s\right\}, \tag{6}$$

where S, ϵ, L, and τ_s denote the set of safe observations, the tolerance for noise, L-Lipschitz constant, and the pre-specified safety threshold, respectively.

This is because, under the Lipschitz continuity, the change in observation with distance is limited. Therefore, as shown in Fig. 2, the observation value of each point can be predicted according to the distance away from the observation of the current safe set. If these predicted observations are confirmed to be safe, they can be included in the safe set, thereby expanding it.

By repeating this process T step, a set of samples predicted to be safe after the T step based on the current safe set S. This is called the T-step ϵ-reachability operator [13] and can be defined as follows:

$$R_\epsilon^T(S) := \underbrace{R_\epsilon(R_\epsilon(\ldots(R_\epsilon(S))\ldots))}_{T \text{ times}}. \tag{7}$$

Safe optimization refers to the process of finding the optimal point in the increased safe set after performing the process of gradually increasing the safe sets by repeating the sampling based on the safety prediction.

As with other safe BO methods [13], [14], we modify our optimization goal as (8)

$$\mathbf{x}^* = \underset{\mathbf{x} \in R_\epsilon^T(S_0)}{\text{argmax}} f(\mathbf{x}), \tag{8}$$

where S_0 is an initial safe seed.

F. REGRET

Instantaneous regret(r_t) is the difference between the global optimum and the actual observation. This is expressed in (9).

$$r_t = f(\mathbf{x}^*) - f(\mathbf{x}_t), \tag{9}$$

where $\mathbf{x}^*$ is the global optimum

In addition, the accumulated value up to the T step is called cumulative regret and is expressed as (10).

$$R_T = \sum_{t=1}^{T} \left(\overbrace{f(\mathbf{x}^*) - f(\mathbf{x}_t)}^{r_t}\right). \tag{10}$$

As the optimization process proceeds, the global optimum is reached, and the observation is close to $f(\mathbf{x}^*)$ and r_t is close to zero. Accordingly, the increased amount of R_T gradually approaches 0 and has a sublinear form. Therefore, we assess the convergence of the algorithm by the sublinearity in the cumulative regret over T iterations. R_T is sublinear as $\lim_{t\to\infty} R_T/t=0$ if the algorithm converges to optimal values given sufficient time. We denote this desirable asymptotic property as *no-regret*, where r_t is strictly positive for $t > 0$ and $f(\mathbf{x}_t)$ is sufficiently close to optimal [9]. Note that r_t or R_T are not revealed to the algorithm. We accumulate the instantaneous regret r_t or the loss from evaluating the function at $\mathbf{x}_t$ instead of at the *a priori unknown* optimal inputs.

III. RELATED WORK

Many Bayesian optimization methods model the underlying expensive black box function with $\mathcal{GP}$ as they effectively provide prior knowledge and model variance under the outlined regularity assumptions. In this section, we review the limitations of such approaches widely used in many successful applications [1], [2], [5], [6], [7], [8].

The curse of dimensionality [27] is a well-known problem in BO applications where the learned function prior requires an exponentially larger amount of data with each increased input dimension. Many BO methods have been proposed to mitigate this problem by making structural assumptions on the objective [28], [29] and decomposing the problem into low dimensional active subspaces using the divide-and-conquer approach [30], [31]. The simplified functional spaces resulting from linear dimensionality reduction are often restricted to linear subspaces of the original domain. In a nonlinear feature space, higher compression rates are achievable [30], and methods such as generative model [32], [33], [34] have been successful in learning the appropriate latent codes and embeddings for low dimensional data representations. However, these methods require a large amount of prior data in offline environments, despite having a relaxed computational cost, even in high dimensional spaces [35], [36]. Other approaches include random forests [37], a heuristic method [38], and an evolutionary algorithm [39] in which particle swarm optimization (PSO) [40] helps realize a computationally efficient optimization process. PSO stochastically approaches the best evaluation point for both each individual particle and the entire population.

Increasing the input dimension causes the number of required data points for effective global convergence to grow, as well as the resulting computations. For GP-based BO methods, growing computations due to the inverse $(\mathbf{K}+\mathbf{I}\sigma^2)^{-1}$ and determinant in the log likelihood $|\mathbf{K}+\mathbf{I}\sigma^2|$ allow for up to a few thousand evaluations in practice [41]. In dealing with this computational bottleneck, methods that leverage parallelism, *e.g.*, using an ensemble of additive $\mathcal{GP}$ models in a randomized divide-and-conquer fashion [11], have shown to effectively address these scalability issues for large-scale observations.

In addition to the growing dimensionality, prior knowledge of optimal kernel hyperparameters also contributes to the increase in computational cost of the optimization problem, which reduces practicality. For $\mathcal{GP}$, it is crucial to set appropriate hyperparameters in advance so as not to fall into a poor local optimum; otherwise, it requires a significant number of data to find the optimal values prior to execution. Since it is difficult to pre-specificy optimal hyperparameters in advance, it is useful to estimate these hyperparameter values in an online manner [23], [25]. These methods guarantee that even if they are not known in advance, they are adjusted at each iteration in the direction of expanding the function space, and finally converge to the global optimum. In safety-critical scenarios, hyperparameter values must be estimated so that the evaluation stays above the safety threshold and safe exploration and exploitation is well-balanced in a global scope.

Safe exploration in muti-armed bandit problems have recently been considered in practical applications in the clinical domain [14], [42], where the problem is constrained by some pre-specified safety threshold. These approaches restrict the optimization goal by defining a ϵ-reachable maximum given a safe seed. While they theoretically guarantee global convergence and yield sufficiently safe performance, their exploration and exploitation strategies by maintaining respective sample sets often yield a poor local optimum as they overlook global exploration in considering safety. In this light, we seek to have a heuristic criterion by which we adjust the kernel hyperparameters in an online manner to safely search globally and escape the local optimum.

Algorithm 1 A-SafeBO
```
Input: Search space $\mathcal{X}$,
       Initial $n$ safe observations
       $O = \{(o_i^{(x)}, o_i^{(y)})\}_{i=0}^{n-1}$,
       Noise $\rho$,
       Safety threshold $\tau_s$,
       User-defined constants $\tau_d$, $\tau_w$
Result: Recommended optimal point, $x^*$
Initialize: $\overline{\mathcal{X}}$, $\overline{O^{(x)}}$, $MB_0$, $S_0$, $r_{\overline{S}}$, $\mathbf{D_0}$
       $\mathcal{GP}$ hyperparameters $(l_0, \beta_0^{1/2}, \mathbf{B}_0)$
       $\overline{x}_0 = x | \exists x \in \overline{O^{(x)}}$
for $t = 1, 2, \ldots, T$ do
    if $t > 300$ then
        Calculate $P_{MB_{t-1}}$ (14)
        $\hat{f} \leftarrow$ Ensemble $\mathcal{GP}(MB_{t-1}, P_{MB_{t-1}})$
    else
        $\hat{f} \leftarrow \mathcal{GP}(MB_{t-1})$
    end
    $\overline{x}_t \leftarrow$ SafePSO$(MB_{t-1}, S_{t-1}, \mathbf{D}_{t-1}, r_{\overline{S}})$
    $x_t \leftarrow$ Denormalize $\overline{x}_t$
    $y_t \leftarrow$ Sample $x_t$ from the real environment
    $MB_t \leftarrow MB_{t-1} \cup \{(\overline{x}_t, y_t)\}$
    Analyze the state of the process (12) and (13)
    Hyperparameters scaling (28)
end
$(x^*, y) \leftarrow \mathrm{argmax}_{(x,y)\in MB}\, y$
```

IV. METHODOLOGY

Given a high evaluation cost and a safety constraint, finding a safe maximizer at a minimal cost is advantageous. In a low budget setting, however, optimization can only be performed in a limited search range due to insufficient sample points for uncertainty estimation, leading to performance degradation [19]. Moreover, hyperparameter misspecification and safety constraints often induce a higher probability of being trapped in a local optimum due to limited exploration.

To remedy these problems, A-SafeBO provides an approach to widen the exploration and computational cost reduction with a new acquisition function, a method for scheduling the hyperparameter scaling process with a limited number of samples, an Ensemble Gaussian Process and a Safe Particle Swarm Optimization. Our method seeks to estimate the objective f based on ρ-sub-Gaussian noise perturbed evaluations using the predictive uncertainty from the $\mathcal{GP}$, similar to [13] and [14]. In addition, it promotes safe exploration in the design space $\mathcal{X}\in\mathbb{R}^d$ so that the acquired samples exceed some pre-specified threshold. Safe exploration is guaranteed by maintaining an increasing sequence of subset $S\subseteq\mathcal{X}$ containing safe samples. The pipeline that shows the operation of our proposed A-SafeBO is shown in Fig. 3.

A-SafeBO finds the optimal point in the safe set by gradually expanding the safe set similar to the algorithm considering safety based on BO. For this, a set of safe observations $(O=\{(o_i^{(x)}, o_i^{(y)})\}_{i=0}^{n-1}$, where $o_i^{(x)}$ is the position information

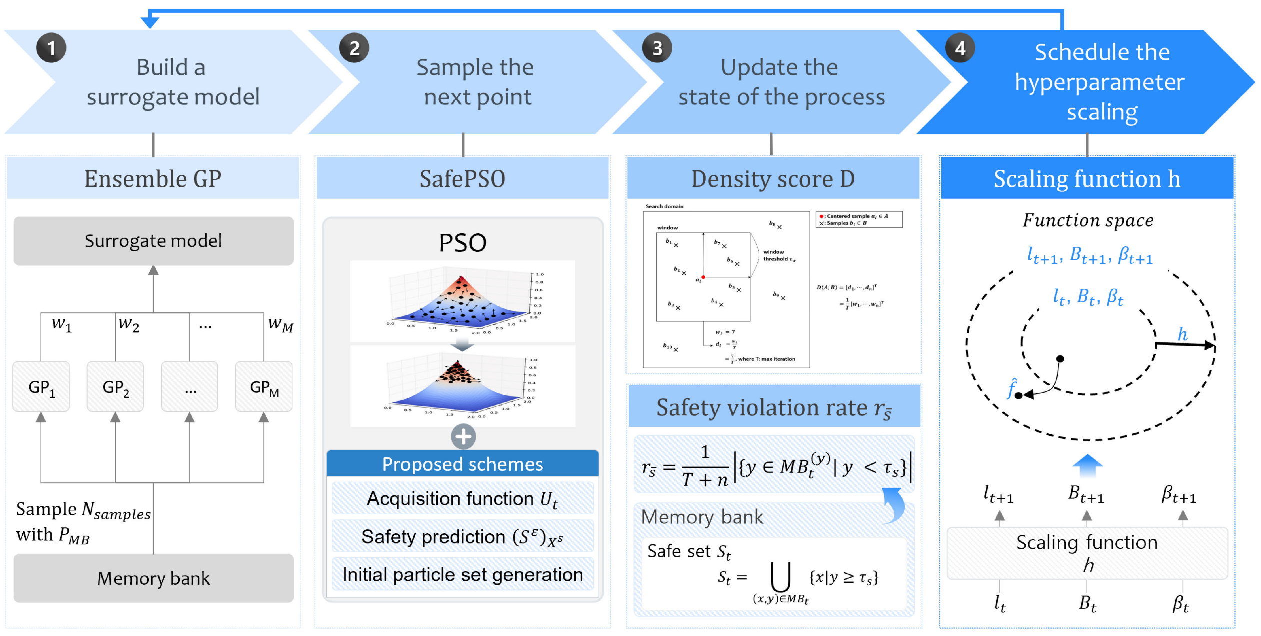
1 Build a surrogate model
2 Sample the next point
3 Update the state of the process
4 Schedule the hyperparameter scaling
Ensemble GP
Surrogate model
w_1 w_2 … w_M
GP$_1$ GP$_2$ … GP$_M$
Sample $N_{samples}$ with P_{MB}
Memory bank
SafePSO
PSO
Proposed schemes
Acquisition function U_t
Safety prediction $(S^{\varepsilon})_{X^S}$
Initial particle set generation
Density score D
Safety violation rate $r_{\bar{S}}$
$r_{\bar{S}} = \frac{1}{T+n}\left|\{y \in MB_t^{(y)} \mid y < \tau_s\}\right|$
Memory bank
Safe set S_t
$S_t = \bigcup_{(x,y)\in MB_t} \{x \mid y \geq \tau_s\}$
Scaling function h
Function space
$l_{t+1}, B_{t+1}, \beta_{t+1}$
l_t, B_t, β_t
h
$\hat{f}$
Scaling function h
l_t B_t β_t

FIGURE 3. The pipeline of the proposed A-SafeBO algorithm.

and $o_i^{(y)}$ is the evaluation result) containing $n(n \geq 1)$ safe samples must be given at the start of the algorithm. At this time, safe means that the evaluation of the sample is higher than the safety threshold τ_s. For notations, we use a superscript in parentheses to represent the set of stored values in O for the respective variable, *i.e.*, $O^{(x)}=\{o_i^{(x)}\}_{i=0}^{n-1}$.

A-SafeBO stores the samples collected during the optimization process in the memory bank (MB) together with the samples stored in this initial safe set O. It then uses them to check the state of the optimization process (*Density score* and *Safety statistics*).

Before explaining how A-SafeBO works, some important concepts must first be explained.

1) THE SEARCH SPACE IS NORMALIZED

A-SafeBO performs an algorithm by normalizing the search space to a unit hypercube ($\bar{\mathcal{X}}$) having a size of 1 in all dimensions to perform optimization regardless of the size of the search space ($\mathcal{X}$). Therefore, the algorithm is performed in a normalized form of all the position information of all samples ($\overline{O^{(x)}}=\{\overline{o^{(x)}}_i\}_{i=0}^{n-1}$ and $\bar{x}_t$, where $\bar{x}_t$ is the normalized position information obtained in the t-th iteration of the optimization process). When the recommended point $\bar{x}_t$ is actually sampled in each optimization step, it is denormalized and sampled in the form of x_t.

2) MEMORY BANK MB

The (MB) stores the normalized positions of the acquired samples and the corresponding evaluations from the real environment, denoted as x and y, respectively, for t iterations.

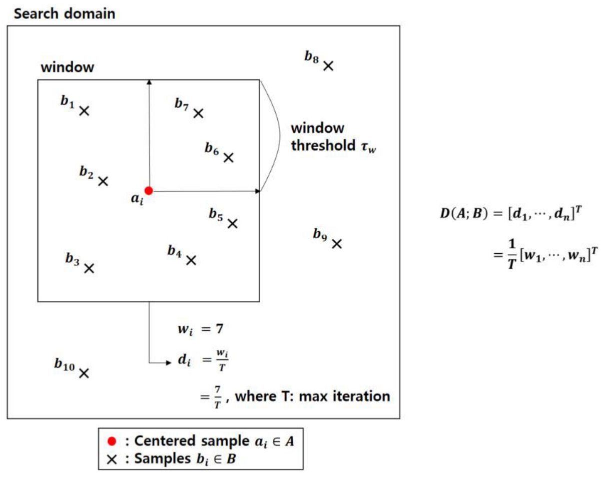
Search domain
window
window threshold τ_w
b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8 b_9 b_{10}
a_i
$w_i = 7$
$d_i = \frac{w_i}{T}$
$= \frac{7}{T}$, where T: max iteration
$D(A;B) = [d_1, \cdots, d_n]^T$
$= \frac{1}{T}[w_1, \cdots, w_n]^T$
● : Centered sample $a_i \in A$
× : Samples $b_i \in B$

FIGURE 4. Example of calculating Density score in 2-dimensional search space.

We update this memory bank at each iteration as (11).

$$MB_t = \{(\bar{o}_i^x, o_i^y) \mid \bar{o}_i^x \in \overline{O^{(x)}}, o_i^y \in O^{(y)}, 0 \leq i \leq n-1\} \bigcup \{(\bar{x}_i, y_i) \mid 0 \leq i \leq t-1\}. \tag{11}$$

The notation representing each variable stored in MB is expressed similarly to that used in O, *i.e.*, $MB_t^{(x)}=\overline{O^{(x)}} \bigcup \{\bar{x}_i\}_{i=0}^{t-1}$.

3) DENSITY SCORE $\mathbf{D}(\cdot)$

Density score ($\mathbf{D}(\cdot)$) is a sample statistic introduced to calculate how densely certain data samples in $\mathbf{A}=\{a_i \in \mathbb{R}^d\}_{i=1}^{n}$

are positioned in the design space with respect to all others belonging to a different set $\mathbf{B}=\{b_i \in \mathbb{R}^d\}_{i=1}^m$. The density score, $\mathbf{D}(\cdot)$, allows the assesment of how populated the acquired samples are in a given window, and becomes useful in iteratively adjusting the safety threshold and choosing the acquisition function. A high $\mathbf{D}(\cdot)$ value for an input data point means that the algorithm is likely to fall into a local optimum with the addition of a new sample point. In (12), the number of samples are counted in the input set $\mathbf{A}$ whose absolute distance is τ_w apart from all the elements in the other set $\mathbf{B}$, and the obtained values are summed and verctorized to find the $\mathbf{D}(\mathbf{A};\mathbf{B}) \in \mathbb{R}^n$.

$$\mathbf{D}(\mathbf{A};\mathbf{B}) = \frac{1}{T}[w_1 \ldots, w_n]^{\mathsf{T}}, \tag{12}$$

with $w_i = \sum_{j=1}^m \prod_{k=1}^d \mathbb{1}(|a_{i,k} - b_{j,k}| \leq \tau_w) \quad \forall i \leq n$, where $\mathbb{1}(\cdot)$ is the indicator function and τ_w denotes a *windowing* constant to control the window size over which we select the samples for comparison. In calculating $\mathbf{D}(\cdot)$, $a_i \in \mathbf{A}$ is compared against all the elements of $\mathbf{B}$ in every dimension in $\mathcal{X}$. For notation purposes, $a_{i,k}$ indicates the value at the k-th dimension of the i-th element in $\mathbf{A}$, and the same applies to $b_{j,k}$. In particular, the density around the t-th sampled point (x_t) is denoted as $\mathbf{D}_t = \mathbf{D}(x_t; MB_{t-1}^{(x)})$.

4) SAFETY STATISTICS

Using the samples stored in MB and the safety thresholds τ_s, we define the safe set, S_t, as a set of samples judged to be safe, and the safety violation rates, $r_{\bar{s}}$, as the proportions of safety-violating samples to the total observations in (13). These statistics are updated on an iteration-basis and allow for a straightforward assessment of which samples in the MB meet the desirable safety requirement.

$$S_t = \bigcup_{(x,y)\in MB_t} \{x | y \geq \tau_s\},$$
$$r_{\bar{s}} = \frac{1}{T+n}\left|\{y \in MB_t^{(y)} | y < \tau_s\}\right| \tag{13}$$

where n is the number of samples in the set of initial safe observations and O and T are the maximum iteration of A-SafeBO.

Using the concepts introduced so far, we can explain how A-SafeBO works in the order of Fig. 3.

A. A SURROGATE MODEL IS BUILT

For safety-constrained BO, we construct a surrogate model based on $\mathcal{GP}$ to precisely estimate the safe sub-region of the search space. Since kernel operation is essential for optimization using $\mathcal{GP}$, the computation time increases exponentially as the number of samples constituting the $\mathcal{GP}$ increases.

However, for a Neural network, as shown in Fig. 1 of [43], the computation time is shown to increase linearly as the number of samples increases. And also, as shown in Fig. 1 of [43], the computational time changes insignificantly with the number of samples when using GP compared to using a neural network, up to approximately 300 samples. We utilize this insight to set the number of samples in each $\mathcal{GP}$ to 300 (*i.e.*, $N_{samples}$=300). Through experiments comparing the changes in computation time for various sample sizes in the Section V-B4, it can be confirmed that setting $N_{samples}$ to 300 is appropriate. Accordingly, we use μ_t and σ_t^2 obtained from the single-model $\mathcal{GP}$ surrogate for a sample size less than or equal to $N_{samples}$. Beyond this sample size, an Ensemble $\mathcal{GP}$ is built to reduce the increase in computation time and make our surrogate more robust, as motivated by the BayesBag algorithm [44]. Our Ensemble $\mathcal{GP}$ consists of $M = \lfloor (n+t)/N_{samples} \rfloor$ (where n is the number of samples in O) $\mathcal{GP}$ models, each of which is constructed from 300 randomly sampled points from the MB without replacement, using the probability vector $P_{MB_{t-1}}$ for $\bar{x}_i \in MB_{t-1}^{(x)}$ $(0 \leq i \leq n+t-1)$ obtained as follows:

$$\begin{aligned}
&P_{MB_{t-1}} \\
&= [P(X = \overline{o^{(x)}}_0), \ldots, P(X = \overline{o^{(x)}}_{t-1}), \\
&\quad P(X = \bar{x}_0), \ldots, P(X = \bar{x}_{t-1})]^{\mathsf{T}} \\
&= \begin{cases} \dfrac{\mathbf{1}_{n+t}^{\mathsf{T}}}{(n+T)}, & \mathbf{D}_{MB} = \dfrac{\mathbf{1}_{n+t}^{\mathsf{T}}}{(n+T)}, \\ \dfrac{1}{n+t}\left[\mathbf{1}_{n+t}^{\mathsf{T}} - \dfrac{(\mathbf{D}_{MB} - \frac{\mathbf{1}_{n+t}^{\mathsf{T}}}{(n+T)})}{\|\mathbf{D}_{MB} - \frac{\mathbf{1}_{n+t}^{\mathsf{T}}}{(n+T)}\|_1}\right], & \text{else} \end{cases}
\end{aligned} \tag{14}$$

where X denotes a random variable and $\mathbf{1}_{n+t}=[1,\ldots,1] \in \mathbb{R}^{n+t}$. $\mathbf{D}_{MB}$ implies $\mathbf{D}(MB_{t-1}^{(x)}; MB_{t-1}^{(x)})$ and $\|\cdot\|_1$ denotes the $L1$-norm. Accordingly, samples with high density scores are assigned smaller probabilities of being sampled. We then repeatedly sample $N_{samples}$ number of samples from MB_{t-1} with $P_{MB_{t-1}}$ to obtain M $\mathcal{GP}$ sub-models at t-th iteration.

To construct an Ensemble $\mathcal{GP}$ using the M $\mathcal{GP}$ sub-models obtained in this way, we aggregate the results from each $\mathcal{GP}$ sub-model. At this time, rather than aggregate each $\mathcal{GP}$ sub-model with the same weight, if the reliability of the built $\mathcal{GP}$ sub-model is higher, a higher weight is given and aggregated. These weights are given based on the reliability calculated by cross-validating each $\mathcal{GP}$ sub-model. Cross validation is performed by calculating the difference between the samples used to build the $i+1$-th sub-model and the prediction obtained using the i-th sub-model as the mean square error (MSE) to check the reliability of the i-th sub-model, where $i = 1, \cdots, M-1$. At this time, to check the reliability of the M-th sub-model, the samples used to build the 1-st sub-model are used. Using the values representing the reliability of each sub-model calculated in this way, the weight to be used to construct the Ensemble $\mathcal{GP}$ is calculated as in (15).

$$\begin{aligned}
w_i &= \frac{\sqrt{(Y_{i+1} - \mu_i(X_{i+1}))^2}}{N_{samples}}, \ (For\ i = 1, \cdots, M-1) \\
w_M &= \frac{\sqrt{(Y_1 - \mu_M(X_1))^2}}{N_{samples}}, \\
\overline{w}_i &= \frac{1 - w_i / \sum_{j=1}^M w_j}{m-1}, \ (For\ i = 1, \cdots, M).
\end{aligned} \tag{15}$$

Using each $\mathcal{GP}$ sub-model, we estimate the mean and standard deviation by weighted aggregating the expected values over all $\mathcal{GP}$ sub-models in (16):

$$\mu_t(x) \approx \sum_{i=1}^{M} \overline{w}_i \cdot \mu_i(x)$$
$$\sigma_t^2(x) \approx \sum_{i=1}^{M} \overline{w}_i \cdot \sigma_i^2(x) + \overline{w}_i \cdot (\mu_i(x) - \mu_t(x)), \tag{16}$$

where $\mu_i(\cdot)$ and $\sigma_i^2(\cdot)$ are the mean and variance, respectively, from the i-th $\mathcal{GP}$ sub-model, $\mathcal{GP}_i$ (i=1, . . . , M). X_i and Y_i are the set of x and y used to construct $\mathcal{GP}_i$

B. THE NEXT POINT IS SAMPLED

The next step is to determine the next point to be sampled by appropriately determining the trade-off between exploration and exploitation with the safety constraint during the search process. At this time, the criteria should be provided to properly adjust the trade-off between exploration and exploitation according to the situation. The BO uses the acquisition function to provide these criteria to determine the next point to be sampled. As discussed in Section III, BO-based $\mathcal{GP}$ methods often face the problem of exponentially increased computing time in the acquisition step particular in an environment with high dimensional data. This problem is mainly due to the use of greed search to optimize the acquisition function to select the next point. To solve this problem, A-SafeBO uses modified Particle Swarm Optimization(PSO) [40] to optimize the acquisition function in a similar way to [39]. This reduces the increase in computation time as the input dimension increases. In this paper, we define the PSO process used to optimize the acquisition function as SafePSO.

1) SAFE PARTICLE SWARM OPTIMIZATION SafePSO

The next point is selected by optimizing acquisition function $U_t(\cdot)$ using SafePSO in each iteration. SafePSO is similar to the existing PSO [40] except that safety is considered in the process of updating particles. This is similar to SafeOPT using PSO [39], except that [39] has three independent swarm optimizers for each of its three acquisition types (maximizer, expander and greed) and A-SafeBO requires only one swarm optimizer. Therefore, it can effectively reduce the computation time compared to [39] in optimizing the acquisition function.

In SafePSO, X^s is the position set of particles that make up the swarm, and V^s is a set of values defining each particle's position change at every iteration. Z^s is a set that stores the position that maximized the acquisition function while satisfying safety until each particle is updated t times and $\overline{z}_t$ is the position that maximizes the acquisition function among the positions stored in Z^s.

SafePSO begins exploration from an initial set of particles(X^s) selected according to the situation. And at each iteration of SafePSO, only the particles that maximize U while satisfying the safety prediction in (27) are updated. The

Algorithm 2 SafePSO

```
Input: Particle Positions $X^s$, Memory Bank $MB$,
       Safe set $S$, Density score $\mathbf{D}$, Unsafe rate $r_{\overline{s}}$
       Surrogate $\mathcal{GP}\ \hat{f}$, Acquisition Function $U$,
       User-defined constants $\tau_s$, $c_1$, $c_2$
Result: Recommended next point, $x^*$
Initialize: Velocities $V^s$,
            Local best positions $Z^s$,
            Global best position $\overline{z}_0$
Set Acquisition function $U_t$ (17), (18), (24)
Select initial set of particles $\mathbf{x}^s$
for $t = 1, \dots, T_{swarm}$ do
    Sample $(b_1)_t \in [0, c_1]$, $(b_2)_t \in [0, c_2]$,
    $\alpha_t \sim \mathcal{U}(0.5, 1)$
    for $i = 1, \dots, N_{swarm}$ do
        $v^i \leftarrow \alpha_t v^i + (b_1)_t(z^i - x^i) + (b_2)_t(\overline{z}_{t-1} - x^i)$
        $x^i \leftarrow x^i + v^i$
    $V^s \leftarrow \{v^1, \dots, v^{N_{swarm}}\}$
    $X^s \leftarrow \{x^1, \dots, x^{N_{swarm}}\}$
    Update $(S^\varepsilon)_{X^s}$
    for $i = 1, \dots, N_{swarm}$ do
        if $x^i \in (S^\varepsilon)_{X^s}$ and $U(x^i) > U(z^i)$ then
            $z^i \leftarrow x^i$
        else
            $z^i \leftarrow z^i$
    $Z^s \leftarrow \{z^1, \dots, z^{N_{swarm}}\}$
    $\overline{z}_t \leftarrow \mathrm{argmax}_{x \in Z^s} U(x)$
Update $(S^\varepsilon)_{Z^s}$
$\overline{z}^* \leftarrow \mathrm{argmax}_{x \in Z^s \cap x \in (S^\varepsilon)_{Z^s}} U(x)$
if $\overline{z}^* = \emptyset$ then
    $x^* \leftarrow \mathrm{argmin}_{x \in S} \mathbf{D}(x; MB^{(x)})$
else
    $x^* \leftarrow \overline{z}^*$
```

SafePSO algorithm returns a recommended point if the final selected particle satisfies S^ε; otherwise, it returns an empty set when there is no sample that satisfies the safety constraint. If an empty set is recommended as the next point by SafePSO, $\mathbf{D}(\cdot)$ for the elements in the safe set against those stored in the MB are calculated, among which the element with the lowest density score is recommended as the next point. This density score-based criterion for acquisition allows for a more reliable surrogate model since it induces further exploration in regions of high uncertainty and a high likelihood of safe yet unexplored samples. The final recommended x_t is then iteratively stacked in the MB.

For SafePSO to work properly, an initial set of particles selected(X^s) according to the situation, an acquisition function(U), and safety prediction(($S^\varepsilon)_{X^s}$) must be given. Therefore, how these are selected in SafePSO according to the circumstances (safety considerations($r_{\overline{s}}$) and whether they fall into the local optimum(D)) will be explained as follows.

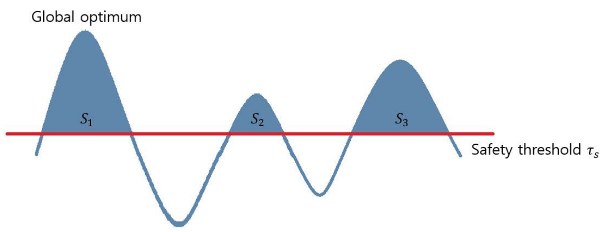

FIGURE 5. Limited explorable area due to safety constraints S_1, S_2, and S_3.

2) ACQUISITION FUNCTION U

The acquisition function is a function that provides a criterion for determining the next point to be sampled as described in Section II. The next point is determined as a point to optimize this acquisition function, so that exploration and exploitation are properly traded off during the optimization process according to the situation.

A-SafeBO performs optimization within a limited number of samples(the maximum iteration, T). A-SafeBO, similar to StageOPT [14], divides T into two phases: **Phase 1** and **Phase 2**. In **Phase 1**, A-SafeBO only focus on expanding the safe set, and in **Phase 2**, A-SafeBO focus on finding the optimal point in the safe set expanded through **Phase 1**. As the goal of the optimization process is different in the two phases, the acquisition function U, which is the criterion for determining the next point in each phase, also has a different form. When the form of the U_t in each phase is defined as U_{p1} for **Phase 1** and U_{p2} for **Phase 2**, U_t is expressed as (17).

$$U_t = \begin{cases} U_{p1}, & t < T \cdot (1-\gamma), \\ U_{p2}, & \text{else}, \end{cases} \tag{17}$$

where T is the maximum iteration of A-SafeBO, and γ is the ratio of the number of samples to be used for **Phase 2** among the total number of samples used for optimization.

Most of the optimization is focused on widening the safe set and then using a small number of samples to find the optimal point inside of it; therefore, γ is mainly a small number.

Phase 1. aims to broaden the safe set. Optimizations with safety constraints have a common goal of expanding the current safe set and finding an optimal point therein. However, since exploration is heavily constrained by safety constraints, the possibility of reaching the global optimum greatly depends on where the initial point is selected.

Fig. 5 shows explorable areas S_1, S_2, and S_3 that satisfy the safe constraint. In this figure, the safety constraints indicate that the evaluation of observations must be higher than the safety threshold τ_s. If the initial point where safe optimization starts is located in S_2 or S_3, no matter how much the optimization process proceeds due to safety constraints, it does not reach the S_1 region where the global optimum exists. Therefore, the global optimum cannot be reached, resulting in poor performance due to the location of the initial point. A-SafeBO proposes FNS, which is a form of U_{p2}, to recognize the limit due to the influence of the initial point position and reduce the limit of exploration due to safety constraint.

For this, it is first necessary to determine whether the exploration of the optimization process is limited. When exploration is limited, the previously collected samples are gathered around the last sampled point. These features are used to determine whether the exploration of the optimization process is in a limited state.

$\mathbf{D_{t-1}}$ is the value calculated by using (13) to find the density of the sample stored in MB_{t-2} around x_{t-1}. A large $\mathbf{D_{t-1}}$ means that there are many previously collected samples around x_{t-1}, which is the most recently acquired sample. In A-SafeBO using $\mathbf{D_{t-1}}$, if this value is greater than the predefined value density threshold (τ_d), the exploration of the optimization process is judged to be limited.

In **Phase 1**, the acquisition function U_{p1} is determined as (18).

$$U_{p1} = \begin{cases} FNS(X^s), & r_{\bar{s}} \le \delta \text{ and } \tau_d \le \mathbf{D_{t-1}}, \\ \underbrace{FNS(X^s)}_{\mathbf{D_{t-1}}} \text{ or } \underbrace{EXP(X^s)}_{1-\mathbf{D_{t-1}}}, & \text{else} \end{cases} \tag{18}$$

where $X^s = \{x_i \in \mathbb{R}^d\}_{i=1}^{N_{swarm}}$ is a set of N_{swarm} particle positions, $r_{\bar{s}}$ is the safety violation rate, δ is the probability of a sample located outside the confidence bound [45], [46], τ_d is the density threshold, and $\mathbf{D_{t-1}}$ is the density score calculated as $\mathbf{D}(x_{t-1}; MB_{t-2}^{(x)})$

FNS is an abbreviation of "Find the New Safe seed", meaning an expression configured to induce broader exploration when exploration is judged to be limited, and EXP is an abbreviation of "Expand", meaning an expression configured to expand the safe set. Both play a role in expanding the safe set, but FNS minimizes safety considerations for a wider exploration, and EXP prioritizes safety constraints and considers exploration and exploitation together.

U_{p1} basically has the form of EXP. When $\mathbf{D_{t-1}}$ is greater than τ_d, it is judged that exploration is limited, so that U_{p1} has the form of FNS. However, since safety considerations should be given the highest priority, even if it is judged that exploration is limited, if $r_{\bar{s}}$ is greater than δ, it is judged that safety considerations are not performed well, and FNS, which lacks safety considerations, is selected as U with low probability. Using $\mathbf{D_{t-1}}$ as this probability, the higher the $\mathbf{D_{t-1}}$, the higher the probability of being selected. However, since $\mathbf{D_{t-1}}$ usually has a small value, FNS is less likely to be selected when the safety violation rate $r_{\bar{s}}$ is high. The reason for using δ as a criterion for safety considerations is as follows. If the prediction for the sample is lower than the lower confidence bound (LCB), it is predicted as unsafe. Therefore, unsafe indicates that the sample exists outside the confidence bound. When modeling based on $\mathcal{GP}$, the probability that the sample exists outside the confidence bound is defined as δ(Appendix A, Lemma 1). When $r_{\bar{s}}$ exceeds δ, the assumption for modeling is broken; therefore, this is used as a criterion that the unsafe rate should not exceed.

To define *FNS* and *EXP*, three functions should be defined first: a margin function($m(X^s)$), Penalty function($P(\alpha, X^s)$), and trade-off function($T_{EE}(X^s)$).

$m(X^s)$ is an expression representing a numerical value indicating how much margin the prediction for the sample has from the safety constraint standard.

$$m(X^s) = LCB_t(X^s) - \tau_s \cdot \mathbf{1}^{\mathsf{T}}_{N_{swarm}}, \tag{19}$$

where $LCB_t(X^s) = \mu_t(X^s) - \beta_t^{1/2} \cdot \sigma_t(X^s)$ is the lower confidence bound computed using Ensemble $\mathcal{GP}$ for X^s.

If the $m(X^s)$ shows a value greater than 0, this indicates that the prediction for the sample is more likely to satisfy the safety constraint. Conversely, if this value is less than 0, the prediction is highly likely to be unsafe.

We can use this $m(X^s)$ to configure the penalty function $P(\alpha, X^s)$, which gives a penalty when the samples is predicted to be unsafe, as in (20).

$$P(\alpha, X^s) = sig(\alpha \cdot m(X^s)), \tag{20}$$

where $sig(\alpha \cdot x) = \frac{1}{1+e^{-\alpha \cdot x}}$ is a sigmoid function that acts as a smooth step function. The $sig(x)$ becomes closer to the shape of the step function as the α value increases. Therefore, as the value of α increases, when the value of x is smaller than 0, it represents a value closer to 0, and when x is larger than 0, it represents a value closer to 1.

If the margin function is less than 0, the $P(\alpha, X^s)$ gives weights close to 0 to the corresponding particles, thereby giving penalties that lower the possibility of being selected as the next point.

And finally, $T_{EE}(X^s)$ is a function that adjusts the trade-off between exploration and exploitation. As described in Section II, the BO performs optimization by newly sampling the next point by appropriately adjusting the trade-off between exploration and exploitation based on the information obtained from the collected samples. Then, the information obtained from the collected samples is expressed through the mean function(μ) and standard deviation function(σ) obtained from the $\mathcal{GP}$. The μ is mainly used to induce exploitation and the σ is used to induce exploration. Similarly in T_{EE}, the μ_t obtained from the Ensemble $\mathcal{GP}$ is used to induce exploitation, and the σ_t obtained from the Ensemble $\mathcal{GP}$ is used to induce exploration as (21).

$$W(\mathbf{D_{t-1}}, r_{\bar{s}}) = sig(T \cdot (\delta - r_{\bar{s}})) \cdot sig(T \cdot (\mathbf{D_{t-1}} - \tau_d)), \tag{21}$$

where T is the maximum iteration of A-SafeBO.

In (21), the density score $\mathbf{D_{t-1}}$ and safety violation rate $r_{\bar{s}}$ are used to generate weights $W(\mathbf{D_{t-1}}, r_{\bar{s}})$ that adjust the trade-off between exploration and exploitation. If $\mathbf{D_{t-1}}$ is higher than the density threshold τ_d and it is judged that the exploration of the optimization process is limited, a larger weight is given. However, since the safety constraint has the highest priority, we use δ and $r_{\bar{s}}$ to determine if the safety consideration is performing adequately. If it is not, it is suppressed from giving a large weight to exploration.

Using this $W(\mathbf{D_{t-1}}, r_{\bar{s}})$, $T_{EE}(X^s)$ is constructed as (22). When safety is considered adequate, if it is judged that exploration of the optimization process is limited, a large weight is given to σ_t (exploration), otherwise a large weight is given to μ_t (exploitation). However, if it is judged that safety is not adequately considered, $W(\mathbf{D_{t-1}}, r_{\bar{s}})$ is given to induce exploitation. Then, since there are often cases where the scales of μ_t and σ_t are different, $\overline{\mu}$ and $\overline{\sigma}$, which are normalized forms, are used.

$$\begin{aligned} T_{EE}(X^s) = &(1 - W(\mathbf{D_{t-1}}, r_{\bar{s}})) \cdot \overline{\mu_t}(X^s) \\ &+ W(\mathbf{D_{t-1}}, r_{\bar{s}}) \cdot \overline{\sigma_t}(X^s). \end{aligned} \tag{22}$$

Based on these functions, $FNS(X^s)$ and $EXP(X^s)$ are defined as (23).

$$\begin{aligned} FNS(X^s) &:= P(1, X^s) \cdot \sigma_t(X^s) \\ EXP(X^s) &:= P(T \cdot r_{\bar{s}}, X^s) \cdot T_{EE}(X^s), \end{aligned} \tag{23}$$

FNS minimizes safety considerations to induce the exploration and uses only the density of surrounding samples using σ_t as a criterion for selecting the next point. Conversely, *EXP* gives a stronger penalty to the point judged to be unsafe as the safety violation rate $r_{\bar{s}}$ is higher, and it induces the safe set to be widened by appropriately adjusting exploration and exploitation according to the situation information obtained using $W(\mathbf{D_{t-1}}, r_{\bar{s}})$.

Phase 2. aims to find the optimal point in the safe set expanded in **Phase 1**. In **Phase 2**, the acquisition function U_{p2} is defined as (24)

$$U_{p2} = MAX(X^s) \tag{24}$$

MAX is an abbreviation of "Maximize", and plays a role in finding the optimal point in the safe set.

Similar to *FNS* and *EXP*, *MAX* is defined as (25) using the functions defined above.

$$\begin{aligned} MAX(X^s) := &P(T \cdot r_{\bar{s}}, X^s) \\ &\cdot (\mu_t(X^s) - \min(\mu_t(X^s))) \end{aligned} \tag{25}$$

As with *EXP*, the higher the safety violation rate $r_{\bar{s}}$ through $P(\cdot)$, the stronger the penalty is given to the point predicted to be unsafe. The optimal point in the safe set is simultaneously found by determining the point with the maximum $\mu_t(\cdot)$ inducing exploitation as the next point.

So far, we have examined the acquisition function U used to determine the next point to be sampled in A-SafeBO. Conventional methodologies have limited exploration due to safety constraints; therefore, their performance is greatly affected by the location of the initial point. However, A-SafeBO can reduce the impact of performance depending on the location of the initial point through *FNS*. This increases the average performance of the algorithm. At the same time, considering the safety violation rate $r_{\bar{s}}$, when the rate of unsafe attempts increases, a stronger penalty for safety is given to induce safer exploration.

3) SAFETY PREDICTION $(S^{\varepsilon})_{X^s}$

In addition to the previously defined U, SafePSO has another criterion for determining the next point: safety prediction $(S^{\varepsilon})_{X^s}$. Safety prediction provides a standard for predicting the safety of particles that can become the next point based on previously collected samples. This safety prediction is determined according to the Lipschitz continuity assumption. According to the Lipschitz continuity assumption defined by (3), the maximum change in the function is bound through the Lipschitz constant L. Using the Lipschitz constant L and the safe set S_{t-1}, safety can be predicted by calculating the one-step ε-reachability operator as (6)

A-SafeBO proposes a standard $SUCB_t(x)$ for predicting the safety of samples using the upper confidence bound of the safe set($UCB_t(x)$ where $x \in S_{t-1}$) based on (3).

$$SUCB_t(x) = \begin{cases} UCB_t(x) + |UCB_t(x)| \\ \cdot (sig(\mathbf{D_{t-1}} - \tau_d) - 0.5), & r_{\overline{s}} < \delta \text{ and } \mathbf{D_{t-1}} > \tau_d, \\ UCB_t(x) - |UCB_t(x)| \\ \cdot (sig(r_{\overline{s}} - \delta) - 0.5), & r_{\overline{s}} > \delta, \\ UCB_t(x), & else, \end{cases} \tag{26}$$

where $x \in S_{t-1}$, $UCB_t(x) = \mu_t(x) + \beta_t^{1/2} * \sigma_t(x)$, $sig(\cdot)$ is a sigmoid function $sig(x) = \frac{1}{1+e^{-x}}$, D_{t-1} is the density function $\mathbf{D}(x_{t-1}; MB_{t-2}^{(x)})$, τ_d is the density threshold, δ is the probability of a sample located outside the confidence bound [45], [46], and $r_{\overline{s}}$ is the safety violation rate

When predicting safety based (6), $SUCB_t(x)$ (26) is used as a criterion to optimistically predict the safety of samples. $SUCB_t(x)$ basically has the value of $UCB_t(x)$ of samples belonging to the safe set $x \in S_{t-1}$. If it is judged that the safety constraint is well satisfied ($r_{\overline{s}} < \delta$) and that the exploration of the optimization process is limited ($\mathbf{D_{t-1}} > \tau_d$), to lower the safety constraint and induce a wider exploration, the $SUCB_t(x)$ is decided by increasing the $UCB_t(x)$ value by a certain part in consideration of $\mathbf{D_{t-1}}$. However, if it is not judged that the safety constraint is well observed ($r_{\overline{s}} > \delta$), safety is considered first, regardless of whether exploration is limited. In this case, to prevent unsafe exploration, $SUCB_t(x)$ is configured by lowering the value of $UCB_t(x)$ considering $r_{\overline{s}}$ to strengthen the safety constraint.

Based on the $SUCB_t(x)$ determined in this way, safety prediction proceeds as (27).

$$(S^{\varepsilon})_{X^s} = \begin{cases} \left\{x' \in X^s \middle| UCB_t(x') \geq \tau_s\right\}, \\ \qquad\qquad U_t = FNS \text{ in } Phase\ 1, \\ \bigcup\limits_{x \in S_{t-1}} \left\{x' \in X^s \middle| SUCB_t(x) - \hat{L} \cdot d(x, x') \geq \tau_s\right\}, \\ \qquad\qquad U_t = EXP \text{ in } Phase\ 1, \\ \left\{x' \in X^s \middle| LCB_t(x') \geq \tau_s\right\}, & else, \end{cases} \tag{27}$$

with $\hat{L} = \max_{(i,j), i \neq j} \left(\frac{|\hat{f}(x_i) - \hat{f}(x_j)|}{d(x_i, x_j)}\right) \ \forall x_i, x_j \in S_{t-1}$ where $d(\cdot, \cdot)$ is an Euclidean distance between two inputs.

The method for safety prediction is classified according to U_t as (27). When U_t is FNS, we minimize safety considerations and aim only to induce a wider exploration; therefore, we optimistically predict safety using UCB_t so that more particles can become the next point. Next, when U_t is EXP, the next sample is determined by adjusting the trade-off between exploration and exploitation according to the situation at every iteration. For safety prediction, $SUCB_t(x)$ is set according to the situation of every iteration, and safety prediction is appropriately adjusted based on this. If the safety constraint is not adequate, a stronger safety constraint is given to reduce attempts on unsafe samples. If it is judged that the exploration of the optimization process is limited when the safety constraint is adequate, the safety constraint is weakened to induce wider exploration. It simultaneously enables more robust safety prediction by estimating the Lipschitz constant $\hat{L}$ using the safe samples accumulated in the safe set S_{t-1}. Finally, when U_t is MAX, it is necessary to find the optimal point only within the safe set; thus, the probability of attempting an unsafe sample is minimized by pessimistically predicting safety based on the LCB_t.

4) AN INITIAL SET OF PARTICLES X^S IS SELECTED

In A-SafeBO, the next point is selected by optimizing U selected according to the situation. Therefore, depending on the purpose of the chosen U, the initial particle set X^s must also be properly selected. When U has the form of FNS and EXP, the purpose of U is to expand the current safe set or to find the seed of a new safe set for a wider exploration. Therefore, samples outside the current safe set should also be considered as initial particles. Therefore, the initial set of particles consists of N_{swarm} number of random samples in a uniform distribution over the entire search space. However, if U is MAX, the purpose of U is no longer to expand the safe set, but to find the optimal point in the safe set. Therefore, X^s must be selected from S_{t-1}. To select X^s, $\mathbf{D}(S_{t-1}; MB_{t-2}^{(x)})$ is calculated for the elements of S_{t-1} and N_{swarm} number of swarm positions are selected, starting with the element with the smallest value. If there are fewer than N_{swarm} elements of S_{t-1}, random sampling with replacement is performed N_{swarm} times from a uniform distribution for all elements of S_{t-1}.

C. THE STATE OF THE PROCESS IS ANALYZED

In A-SafeBO, the next step with the sample is determined by selecting the appropriate acquisition function U, safety prediction $(S^{\varepsilon})_{X^s}$, and initial set of particles X^s according to the status of the current optimization process. As a result, it can induce wider exploration without significantly compromising safety considerations.

To understand the current state of the optimization process, A-SafeBO updates the safe set S, density score $\mathbf{D}$, and safety violation rate $r_{\overline{s}}$ at every iteration in the same way as in (12) and (13) using the sample information stored in the MB.

Using this updated information, if $r_{\overline{S}}$ is less than δ, it is judged that safety consideration is adequate during the optimization process. In addition, if the density score indicating the density of the samples stored in the memory bank near the last sampled point $D_{t-1} = \mathbf{D}(x_{t-1}; MB^{(x)}_{t-2})$ is smaller than the density threshold τ_d set by the user, it is judged that exploration is not restricted.

When it is judged that safety considerations are inadequate, U, $(S^{\varepsilon})_{X^s}$, and X^s are selected to reduce unsafe attempts by limiting exploration as safety considerations are the top priority during the optimization process.

If it is judged that safety considerations are adequate and that exploration is limited, U, $(S^{\varepsilon})_{X^s}$, and X^s are selected to induce exploration. When it is determined that the exploration is not limited, the current D_{t-1} and $r_{\overline{S}}$ are used to appropriately trade off exploration and exploitation to select the next point.

D. HYPERPARAMETER SCALING ARE SCHEDULED

A key component of A-SafeBO is the hyperparameter scaling step over successive iterations, where the kernel hyperparameters are adaptively tuned based on the sampling outcome. Specifically, the algorithm adjusts the lengthscale l, the bounded RKHS norm $\mathbf{B}$, and the exploration-exploitation trade-off factor $\beta^{1/2}$. While common BO-based optimization methods require a hyperparameter tuning step prior to deployment in real scenarios, online hyperparameter scaling in A-SafeBO is more practical with reduced computational cost and time.

The idea of online hyperparameter scaling has been successful and in practical use for its general applicability across various working environments [23], [25]. For instance, A-GP-UCB [25] gradually increases the function space by decreasing l and increasing $\mathbf{B}$ over iterations to escape being trapped in a local optimum and more effectively reach the global optimum, grounded in its theoretical convergence proof. It also ensures global convergence empirically from its regret converging to zero and the cumulative regret having a sublinear form. While A-SafeBO also follows a similar approach, It is differentiated by the following characteristics.

- It can be effectively applied in situations where the number of samples used in the optimization process is limited to a small number.
- it proposes a scaling function h with the form of a half life function.

As with A-GP-UCB, A-SafeBO can better widen exploration by inducing regions of high uncertainty to be explored more frequently. However, A-SafeBO induces exploration by effectively scheduling the hyperparameter scaling process using h in a situation where the number of samples that can be used is limited, unlike A-GP-UCB, where the number of samples is unlimited. Since an infinite number of samples cannot be used in an actual application, these differences increase the practicality of A-SafeBO. We scale l, $\mathbf{B}$, $\beta^{1/2}$ at t-th iteration with a half-life scaling function h as follows:

$$\begin{aligned} l_t &= l_0 \cdot h_t^{1/d} \\ \mathbf{B}_t &= \mathbf{B}_0 / h_t \\ \beta_t^{1/2} &= \max\left(\beta_{t-1}^{1/2}, \mathbf{B}_t + 4\rho\sqrt{I(y_t;\hat{f})+1+\ln(1/\delta)}\right), \end{aligned} \tag{28}$$

with

$$h_t = \exp\left(\frac{-5t\ln 2}{T}\right),$$

$$I(y_t;\hat{f}) = \begin{cases} 0.5\log|\mathbf{I} + \rho^{-2}\mathbf{K}|, & t \leq 300, \\ \sum_{m=1}^{M} 0.5\log|\mathbf{I} + \rho^{-2}\mathbf{K}_m|, & \text{else,} \end{cases}$$

where $\mathbf{I}$, ρ, $\mathbf{K}$, $\mathbf{K}_m$ and $|\cdot|$ are the identity matrix, noise, the kernel matrix $[k(x,x')]_{x,x' \in MB^{(x)}_{t-1}}$, the kernel matrix $[k(x,x')]_{x,x' \in \mathcal{GP}^{(x)}_m}$ and the determinant operator, respectively. For $t > 300$, $I(y_t;\hat{f})$ is computed as a sum over all mutual information of all $\mathcal{GP}$ sub-models. To apply hyperparameter scaling irrespective of the size of the input space, A-SafeBO initially scales the input space to a unit hypercube with a size of 1 over all dimensions. Since space is normalized to 1, we initialize l_0=1, $\mathbf{B}_0$=1, and h_0=1. The theoretical proofs for the extension of function space and the convergence of A-SafeBO are shown in Appendix A Lemma 2 and Theorem 1.

For evaluating an expensive objective function in practice, the scaling function h ensures that the bounded RKHS norm $\mathbf{B}$ monotonically grows and the lengthscale l monotonically decreases at adequate rates such that the function space is widened during the search, yet the sublinear regret form is visible given a finite number of iterations.

Here, we have introduced the process of finding the optimal point in the safe set by conducting safe exploration through A-SafeBO. Through the introduced methods, A-SafeBO does not significantly impair stability when compared to existing safe exploration methods based on BO, but is more likely to find a better optimal point by inducing wider exploration and reduces computational cost to proceed with a more efficient optimization process. In addition, it efficiently induces exploration by scheduling hyperparameter scaling within a given number of samples without prior knowledge of the kernel hyperparameters. Through this, the resources required to tune the kernel hyperparameters can be saved considerably. These advantages further increase the practicality of A-SafeBO in industrial environments where available resources (e.g. time and number of samples used for optimization) are limited.

V. EXPERIMENTS

In this section, we validate our algorithm on several synthetic benchmark functions and apply it to a real-world optimization problem under a safety threshold constraint to demonstrate performance improvements relative to previous approaches

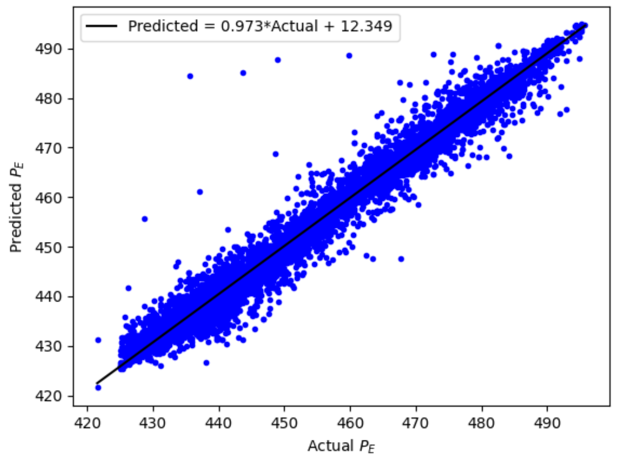

FIGURE 6. **Ensemble bagging decision tree regressor test predictions with 80-20 train-test split (RMSE=2.752). This decision tree is used as the ground truth objective function for the CCPP dataset.**

GP-UCB [9], SafeOpt [13] and StageOpt [14]. The comparison with GP-UCB is conducted to compare the optimization results with and without considering safety, while the comparison with SafeOpt and StageOpt is conducted to evaluate the performance of our algorithm against previous safe BO methods. The code for the experiments is available at https://github.com/KK-Han/A-SafeBO.git.

A. EVALUATION ENVIRONMENT

1) SYNTHETIC BENCHMARK FUNCTIONS

Synthetic benchmark functions from the Virtual library of simulation experiments and an open-source repository [47], [48] were used to evaluate the performance of A-SafeBO. Specifically, we chose Griewank 2-D, Adjiman 2-D, Hartmann 6-D and Periodic 10-D functions, as detailed in Table 1. The functions were negated to formulate as maximization problems and the search space was constrained with the bound B in each dimension.

2) REAL-WORLD APPLICATION

We demonstrate the practical effectiveness of A-SafeBO by using a publicly available industrial dataset: the combined cycle power plant (CCPP) from the UCI repository [49]. In each cycle of CCPP, gas and steam turbines generate electricity, which is then transferred to heat recovery steam generators. Under a fully-loaded functional plant condition, the ambient variables (AT, V, AP, RH) measured from various sensors located around the plant yield the net hourly electrical energy (PE) every second. The CCPP dataset contains five folds of data, each containing 9,568 data instances acquired from a fully-functional plant for 674 days over the span of six years from 2006 to 2011. The dataset was randomly shuffled and an ensemble of bagging regression trees was constructed with 10 base estimators from scikit learn [50]. The data statistics are described in Table 2 and the prediction performance of the ensemble regressor is described in Fig. 6.

This regressor trained on all four ambient variables was taken as the ground truth function for this real-world environment for the following insights: (1) the ground truth function for the power plant can neither be defined or measured in practice with sufficient samples, (2) the prediction accuracy of tree-based learning algorithms have shown to reduce errors compared to other regression methods [51], and (3) averaging in ensemble methods reduces variance. In addition, we note that the number of available samples for the number of given features is sufficient to adequately learn a regression model, as the data are collected over a long period and are thus highly representative of the population [51], [52].

3) EVALUATION METRICS

To compare the performance of algorithms, the following four evaluation metrics were used in the environment for the benchmark function and real world problem.

Reward represents the evaluation value of the point that is finally recommended as the optimal point through each algorithm. Since problems in all environments are unified as a maximization problem, the larger the reward value, the better the performance.

Computation time refers to the time consumed to obtain a recommendation through the algorithm. Since the recommendation was obtained using the same number of samples in all three algorithms, the shorter the computation time, the faster the algorithm.

Safety rate represents the proportion of samples that are actually confirmed to be safe with an evaluation higher than the safety threshold among the total number of samples used for optimization. It can be said that the higher the safety rate, the better the algorithm with safety considerations.

Minimum regret indicates the difference between the global optimum and the evaluation as (9). Among them, the minimum regret means the difference between the recommendation obtained through the optimization algorithm and the global optimum as (29).

$$r_{min} = f(\mathbf{x}^*) - f(\mathbf{x}'), \tag{29}$$

where $\mathbf{x}^*$ is the global optimum and $\mathbf{x}'$ is the optimum estimated through the algorithm.

The closer this value is to 0, the better this algorithm converges to the global optimum.

B. IMPLEMENTATION DETAILS

Our algorithm was experimentally compared with one of the most used BO method, GP-UCB [9] and two widely-used safe BO methods: SafeOpt [13], [53] and StageOpt [14], a complementary version of SafeOpt, on an Ubuntu 18.04, Intel i7-6700k machine with 64 GB RAM. For fair comparison, we used these baselines modified with swarm optimization. In the comparison with GP-UCB, we just replaced the acquisition function in A-SafeBO with the GP-UCB method to solely examine the effect of considering safety (GP-UCB*). The parameter settings for the environments, thresholds and

TABLE 1. Synthetic Benchmark Functions.

Name	Dimension, d	Function To Maximize	Bound, B $\forall i \in [1,\ldots,d]$	Global Optimum, $x^* \in \mathbb{R}^d \rightarrow f(x^*)$
Griewank	2	$f(x) = -\sum_{i=1}^{d} \frac{x_i^2}{4000} + \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) - 1$	$x_i \in$ [-5, 5]	(0, 0) $\rightarrow$ 0
Adjiman	2	$f(x) = -\cos(x_1)\sin x_2 + \frac{x_1}{x_2^2+1}$	$x_i \in$ [-5, 5]	(5, 0) $\rightarrow$ 5
Hartmann	6	$f(x) = \sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{6} A_{ij}(x_j - P_{ij})^2\right),$ where $A = \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}$ and $P = 10^{-4}\begin{bmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{bmatrix}$	$x_i \in$ [0, 1]	(0.2017, 0.1500,0.4769, 0.2753, 0.3117, 0.6573) $\rightarrow$ 3.322
Periodic	10	$f(x) = -1 - \sum_{i=1}^{d} \sin^2(x_i) + 0.1\exp\left(\sum_{i=1}^{d} x_i^2\right)$	$x_i \in$ [-2, 2]	(0,0,0,0,0,0,0,0,0,0) $\rightarrow$ -0.9

TABLE 2. CCPP dataset features statistics.

Variable	Unit	Min	Max	Mean	
Ambient Temp. (AT)	°C	1.81	37.11	19.65 ± 7.45	
Exhaust Vacuum (V)	cm Hg	25.36	81.56	54.31 ± 12.71	
Ambient Pressure (AP)	mbar	992.89	1,033.30	1013.26 ± 5.94	
Relative Humidity (RH)	%	25.56	100.16	73.41 ± 14.60	
Plant Energy (PE)	MW	420.26	496.76	454.37 ± 17.07	
Correlation	**AT**	**V**	**AP**	**RH**	**PE**
AT	1.00	0.84	-0.51	-0.54	-0.95
V		1.00	-0.41	-0.31	-0.87
AP			1.00	0.10	0.52
RH				1.00	0.39
PE					1.00

TABLE 3. Parameter settings for experiments on (a) Griewank 2-D, (b) Adjiman 2-D, (c) Hartmann 6-D, (d) Periodic 10-D and (e) CCPP (GP-UCB*: GP-UCB with Ensemble $\mathcal{GP}$ and hyperparameter scaling).

Dataset	(a)		(b)		(c)		(d)		(e)	
Environment										
N_{swarm}	20		20		60		100		40	
T_{swarm}					100					
ρ					0.01					
Threshold										
τ_s	-0.9		-0.1		1.2		-5		453	
τ_w					0.05					
τ_d					0.05					
Hyperparameters	l	β	l	β	l	β	l	β	l	β
SafeOpt	0.3	2.0	0.3	2.5	0.7	3.0	0.7	3.0	0.2	3.0
StageOpt	0.3	2.0	0.3	2.5	0.7	3.0	0.7	3.0	0.2	3.0
GP-UCB*					Not necessary					
A-SafeBO (Ours)					Not necessary					

GP-UCB*: GP-UCB with Ensemble $\mathcal{GP}$ and hyperparameter scaling

kernels are outlined in Table 3. When using swarm optimization, a higher dimensional search space uses more particles for efficient optimization. Prior to the algorithm, the environment-specific function space is normalized to a value between 0 and 1 in every dimension. For the CCPP data, the ensemble bagging decision tree regressor was taken as the ground truth objective function as it is difficult to test in practice.

1) NUMBER OF SAMPLES USED FOR OPTIMIZATION T

To check the performance and practicality of the algorithm, we assumed two situations and conducted an experiment.

Sufficient number of samples ($T = 1000$) The performance of the algorithm was evaluated when a sufficient number of samples were available for optimization in each environment. In these experiments, we can evaluate the performance of the algorithm based on the evaluation metrics of the algorithm.

Small number of samples ($T = 100$) The performance of the algorithm was evaluated when using only a small number of samples in a situation where the number of samples that can be used for optimization is limited, as in the actual industrial field.

Through these experiments, the performance of the algorithm as well as its practicality can be confirmed.

2) SAFETY THRESHOLD τ_s AND INITIAL SAFE OBSERVATIONS O

Since the exploration in safe BO-based approaches is limited by safety constraints, the optimization performance is greatly affected by the location of the initial point. Through these experiments, our goals was to confirm that A-SafeBO shows superior performance over existing methods by inducing wider exploration in a situation where exploration is limited due to a safety constraint without significantly harming the safety constraint.

To this end, in the 2D environment(Griewank 2D and Adjiman-2D) where the local optimum and the global optimum can be easily distinguished visually, the safety threshold

τ_s is set so that a space that cannot be crossed is created between the local optimum and the global optimum due to safety constraints. To reduce the influence of optimization performance due to the location of the initial point, 10 points with observations higher than the safety threshold were randomly selected where the local optimum is located. These 10 points became the initial safe observation O in common in each of the three algorithms, and the experiment, with each of the 10 points as the starting point, was repeated 10 times. Thus, 100 experimental results were obtained for each algorithm in each environment.

Since the location of the local optimum cannot be visually confirmed in the 4D environment or higher, the safety threshold τ_s cannot be determined in the same way as in the 2D environment. Therefore, in the 4D or higher environment, the safety threshold τ_s was randomly selected between the min and max values of evaluation in each environment. Additionally, to reduce the effect of the initial point on the performance of the optimization, the ten initial points were randomly selected from the search space as points with an evaluation greater than the selected safety threshold τ_s. As in the 2D case, 10 points were set as the initial safe observation O and the optimization process was repeated 10 times using each point selected in each environment as the initial point of the 4 algorithms. Thus, in each environment, all three algorithms obtained 100 experimental results.

The safety threshold τ_s selected in each environment is shown in Table 3. Table 4 and Table 5 summarize the results of the 100 experiments in each environment when the sample used for optimization is sufficient and limited using the mean and standard error.

3) HYPERPARAMETER TUNING

GP-UCB* and A-SafeBO do not require setting kernel hyperparameters prior to the optimization process due to online hyperparameter scaling, whereas SafeOpt and StageOpt require the setting of kernel hyperparameters before the search process for practical deployment. After function space normalization, we set the initial lengthscale, l, to 1 (max), which is successively adjusted in 0.1 increments, and set the initial β to 2, which is then greedily searched from 2 to 3 in 0.1 increments. Note that a β value between 2 and 3 is common by convention. This search process for each environment is repeated ten times, totalling 1,100 times (ten times for each of l, β and environment) in search of the optimal kernel hyperparameters. As for StageOpt, 90% and 10% of the total iterations are used for the safe region expansion phase and the utility optimization phase, respectively. Similarly, A-SafeBO also used a gamma of 0.1, using 90% of the samples in **Phase 1** and 10% of the samples in **Phase 2**.

4) NUMBER OF SAMPLES($N_{samples}$) COMPOSING THE ENSEMBLE $\mathcal{GP}$

A-SafeBO constructs a surrogate model using an Ensemble $\mathcal{GP}$ to address the problem of exponentially increasing computation time of $\mathcal{GP}$-based BO methods as the number of samples increases. In the case of using a neural network to construct the surrogate model, the computation time increases linearly with the number of samples as it increases. As shown in Fig. 1 of [43], up to approximately 300 samples, the increase in computation time is not significantly different between using neural network and $\mathcal{GP}$, so A-SafeBO limits the number of samples in a $\mathcal{GP}$ sub-model ($N_{samples}$) that make up the Ensemble $\mathcal{GP}$ to 300. We investigate the effect of changing $N_{samples}$ on the computation time of A-SafeBO by varying $N_{samples}$ in two environments (Griewank 2-D, Adjiman 2-D) to see if defining $N_{samples}$ as 300 is appropriate.

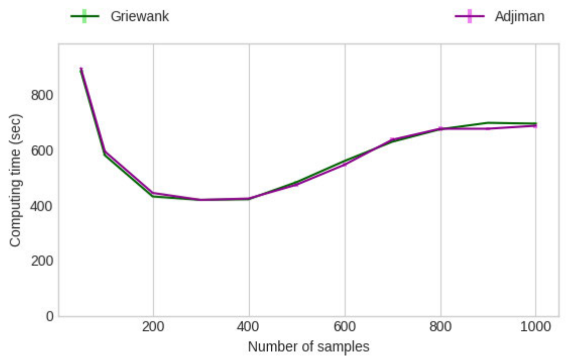

FIGURE 7. The graph represents the change in computation time according to the variation in the number of samples used to construct the $\mathcal{GP}$ sub-models ($N_{samples}$) that make up the Ensemble $\mathcal{GP}$.

As shown in the Fig. 7, it is confirmed that the computation time is the smallest when $N_{samples}$ is set to 300. When $N_{samples}$ is less than 300, the computation time for constructing and operating a single $\mathcal{GP}$ sub-model is low, but since there are many $\mathcal{GP}$ sub-models to be ensemble, it takes a lot of time to estimate the surrogate model, resulting in an increase in computation time. On the other hand, when the number of $N_{samples}$ exceeds 300, it takes a long time to construct and operate a single $\mathcal{GP}$ sub-model, resulting in a longer computation time to estimate the entire surrogate model. Therefore, considering this trade-off, it was confirmed that setting $N_{samples}$ to 300 was appropriate. The current version of A-SafeBO does not parallelize the $\mathcal{GP}$ sub-models, but if all $\mathcal{GP}$ sub-models are fully parallelized, it may be possible to improve computation speed.

C. EXPERIMENT RESULTS AND ANALYSIS

The experimental results are summarized in Table 4 and Table 5. Table 4 shows the experimental result when a sufficiently large number of samples are used for optimization, and Table 5 shows the experimental result when only a small number of samples are used because those available for use are limited.

1) SUFFICIENT NUMBER OF SAMPLES ($T = 1000$)

a: REWARD AND r_{min}

As mentioned earlier when introducing evaluation metrics in Section V-A3, it is possible to check how close the

TABLE 4. Experimental Results on (a) Griewank 2-D (b) Adjiman 2-D (c) Hartmann 6-D (d) Periodic 10-D functions, and (e) CCPP. The results of using the three algorithms were compared through the mean and standard deviation of the results of 100 experiments for the optimization process conducted using sufficient samples ($T = 1000$) in each environment.

Function	Algorithm	Reward ↑		Safety Rate (%) ↑	Time (s) ↓	Minimum regret ↓
		Global	Estimated			
(a)	SafeOpt	0.0000	$-0.2317_{\pm0.0049}$	$80.8400_{\pm0.1870}$	$1149.2337_{\pm0.8956}$	$0.1037_{\pm0.0022}$
	StageOpt		$-0.2069_{\pm0.0049}$	$77.8120_{\pm0.2236}$	$789.4267_{\pm0.6401}$	$0.0819_{\pm0.0020}$
	GP-UCB*		$-0.0134_{\pm0.0054}$	$65.2760_{\pm0.2630}$	$240.5839_{\pm0.2670}$	$0.0781_{\pm0.0034}$
	A-SafeBO (Ours)		$-0.1156_{\pm0.0115}$	$93.1520_{\pm0.0722}$	$381.1532_{\pm0.7072}$	$0.0189_{\pm0.0098}$
(b)	SafeOpt	5.0000	$0.5641_{\pm0.0268}$	$79.2929_{\pm0.3075}$	$1148.0689_{\pm1.0128}$	$4.3128_{\pm0.0269}$
	StageOpt		$0.5657_{\pm0.0280}$	$77.8255_{\pm0.2368}$	$793.2017_{\pm0.5833}$	$4.3026_{\pm0.0278}$
	GP-UCB*		$4.9972_{\pm0.0052}$	$83.0610_{\pm0.0675}$	$239.1366_{\pm0.2302}$	$0.1147_{\pm0.0021}$
	A-SafeBO (Ours)		$4.9637_{\pm0.0412}$	$94.3560_{\pm0.0847}$	$372.9850_{\pm1.1206}$	$0.0929_{\pm0.0400}$
(c)	SafeOpt	3.3223	$2.9889_{\pm0.0150}$	$98.1340_{\pm0.0696}$	$1760.4057_{\pm1.1330}$	$0.2837_{\pm0.0144}$
	StageOpt		$2.8999_{\pm0.0153}$	$98.0010_{\pm0.0686}$	$1198.5388_{\pm0.8408}$	$0.3580_{\pm0.0152}$
	GP-UCB*		$3.1494_{\pm0.0089}$	$31.0210_{\pm0.1344}$	$433.4951_{\pm0.6374}$	$0.1155_{\pm0.0076}$
	A-SafeBO (Ours)		$3.2291_{\pm0.0067}$	$97.8290_{\pm0.0634}$	$929.3716_{\pm5.4128}$	$0.0311_{\pm0.0045}$
(d)	SafeOpt	-0.9000	$-2.4362_{\pm0.0199}$	$85.0350_{\pm0.1357}$	$3098.8831_{\pm2.3754}$	$1.5318_{\pm0.0191}$
	StageOpt		$-1.4390_{\pm0.0153}$	$84.7050_{\pm0.1841}$	$1755.4624_{\pm1.5127}$	$0.5238_{\pm0.0143}$
	GP-UCB*		$-1.2506_{\pm0.0147}$	$27.8880_{\pm0.0896}$	$575.9747_{\pm0.6549}$	$0.3245_{\pm0.0129}$
	A-SafeBO (Ours)		$-1.0545_{\pm0.0110}$	$98.4000_{\pm0.0097}$	$1581.8382_{\pm1.5655}$	$0.1067_{\pm0.0098}$
(e)	SafeOpt	496.7600	$491.3480_{\pm0.1120}$	$90.4710_{\pm0.4341}$	$1589.5184_{\pm3.6784}$	$4.3925_{\pm0.1109}$
	StageOpt		$491.0284_{\pm0.1222}$	$88.7340_{\pm0.5379}$	$1077.1136_{\pm2.4879}$	$4.6961_{\pm0.1212}$
	GP-UCB*		$493.7930_{\pm0.1321}$	$99.0740_{\pm0.1637}$	$410.5731_{\pm0.5933}$	$1.9515_{\pm0.1318}$
	A-SafeBO (Ours)		$492.4861_{\pm0.3868}$	$99.8450_{\pm0.0282}$	$832.2630_{\pm3.8611}$	$3.2585_{\pm0.3875}$

GP-UCB*: GP-UCB with Ensemble $\mathcal{GP}$ and hyperparameter scaling

TABLE 5. Experimental Results on (a) Griewank 2-D (b) Adjiman 2-D (c) Hartmann 6-D (d) Periodic 10-D functions, and (e) CCPP. The results of using the three algorithms were compared through the mean and standard deviation of the results of 100 experiments for the optimization process conducted using sufficient samples ($T = 100$) in each environment.

Function	Algorithm	Reward ↑		Safety Rate (%) ↑	Time (s) ↓	Minimum regret ↓
		Global	Estimated			
(a)	SafeOpt	0.0000	$-0.2034_{\pm0.0051}$	$90.0500_{\pm0.4354}$	$27.7374_{\pm0.0337}$	$0.1320_{\pm0.0027}$
	StageOpt		$-0.2104_{\pm0.0050}$	$89.1200_{\pm0.5324}$	$18.6454_{\pm0.0248}$	$0.1708_{\pm0.0036}$
	A-SafeBO (Ours)		$-0.1830_{\pm0.0077}$	$91.8200_{\pm0.2430}$	$10.2607_{\pm0.0214}$	$0.1161_{\pm0.0061}$
(b)	SafeOpt	5.0000	$0.5770_{\pm0.0279}$	$91.6300_{\pm0.4104}$	$27.8934_{\pm0.0762}$	$4.3604_{\pm0.0286}$
	StageOpt		$0.5690_{\pm0.0282}$	$90.7188_{\pm0.4694}$	$18.6981_{\pm0.0211}$	$4.3707_{\pm0.0268}$
	A-SafeBO (Ours)		$3.1260_{\pm0.2077}$	$92.9400_{\pm0.2440}$	$10.2880_{\pm0.0254}$	$1.8125_{\pm0.2081}$
(c)	SafeOpt	3.3223	$2.7607_{\pm0.0184}$	$91.2929_{\pm0.3030}$	$31.4122_{\pm0.0339}$	$0.5242_{\pm0.0184}$
	StageOpt		$2.5570_{\pm0.0374}$	$94.7653_{\pm0.2674}$	$21.1455_{\pm0.0445}$	$0.7338_{\pm0.0347}$
	A-SafeBO (Ours)		$3.0453_{\pm0.0173}$	$94.1700_{\pm0.1025}$	$14.1883_{\pm0.0500}$	$0.2275_{\pm0.0175}$
(d)	SafeOpt	-0.9000	$-2.4653_{\pm0.0231}$	$63.2400_{\pm0.5881}$	$46.6053_{\pm0.1703}$	$1.5638_{\pm0.0222}$
	StageOpt		$-2.5761_{\pm0.0263}$	$46.7500_{\pm1.4892}$	$30.8512_{\pm0.1435}$	$1.6741_{\pm0.0252}$
	A-SafeBO (Ours)		$-1.5058_{\pm0.0420}$	$83.3200_{\pm0.1449}$	$18.1246_{\pm0.0491}$	$0.5934_{\pm0.0403}$
(e)	SafeOpt	496.7600	$476.1498_{\pm1.2457}$	$98.3300_{\pm0.2207}$	$32.9582_{\pm0.1678}$	$19.6026_{\pm1.2443}$
	StageOpt		$476.3033_{\pm1.2709}$	$98.1300_{\pm0.2356}$	$22.0323_{\pm0.1406}$	$19.4341_{\pm1.2682}$
	A-SafeBO (Ours)		$485.5619_{\pm1.2226}$	$98.0100_{\pm0.3050}$	$13.9890_{\pm0.0949}$	$10.1951_{\pm1.2219}$

recommendation obtained through the algorithm is to the global optimum through Reward and r_{min}.

Table 4 confirms that the A-SafeBO proposed in this paper, except for GP-UCB*, shows the highest reward and the closest r_{min} to 0 in all five environments. This is because GP-UCB* does not consider safety and allows for unrestricted exploration, resulting in a higher probability of reaching the global optimum. These results indicate that the recommendation obtained through A-SafeBO is closest to the global optimum among the three safe BO algorithms.

This is because if A-SafeBO falls into the local optimum, it can be identified using the density score $\mathbf{D}$. If it is judged that the exploration of the optimization process is limited, the next point is determined by effectively adjusting the acquisition function U, safety prediction $(S^{\varepsilon})_{X^s}$, and initial set of particles X^s to induce wider exploration. For these reasons, A-SafeBO shows superior performance than conventional methodologies.

Fig 8 shows the results of optimization using each algorithm in a 2D environment. In these results, all three

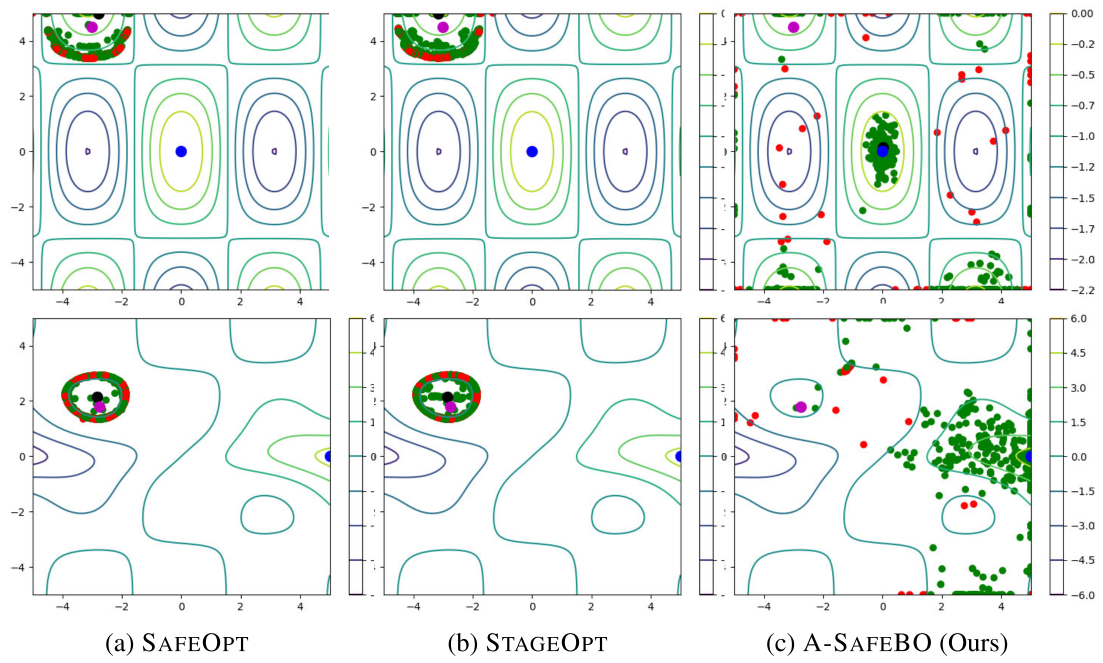

FIGURE 8. Experimental results on the benchmarks (only 2-D functions for visualization). From top to bottom rows: Griewank ($\tau_S = -0.9$), Adjiman ($\tau_S = -0.1$). Points in blue, black, purple, red and green indicate the global optimum, estimated optimum, initial point, unsafe samples, and safe samples, respectively.

algorithms start optimization at the same position near the local optimum. Although it is difficult to reach the global optimum due to the safety constraint limiting exploration, it can be confirmed that A-SafeBO leaves the local optimum and reaches the global optimum. However, SafeOpt and StageOpt do not deviate from the local optimum due to safety constraints.

Fig. 10 and Fig. 9 are graphs showing the changes in r_{min} and cumulative regret obtained through 100 experiments using three algorithms in each environment as optimization proceeds through the mean and standard error.

The graphs in Fig. 10 show that the r_{min} of A-SafeBO is closest to 0 in all environments as the optimization progresses. These results imply that the evaluations of A-SafeBO in all environments are closest to the global optimum among the three safe BO algorithms.

The graphs in Fig. 9 confirm that the cumulative regret of A-SafeBO has the smallest value in all environments and the graph most similar to the sublinear form. This is because the instantaneous regret r_t obtained during the optimization process is closest to 0 among the three safe BO algorithms. This means that the A-SafeBO converges closest to the global optimum.

b: SAFETY RATE (%)

Table 4 demonstrates that the safety rate of A-SafeBO is similar or greater than that of other safe BO algorithms. These results confirm that A-SafeBO induces more exploration compared to other safe BO algorithms without significantly compromising safety considerations. This is achieved through the use of the safety violation rate $r_{\bar{S}}$ to identify cases where the safety constraint is not well satisfied. In such instances, a stronger safety constraint is imposed via safety prediction $(S^{\varepsilon})_{X^s}$ to limit unsafe attempts through exploration reduction using U.

On the other hand, GP-UCB* exhibits lower safety when compared to methods that consider safety. This is due to the fact that GP-UCB* does not incorporate safety considerations, leading to exploration in regions predicted to be unsafe. Although this can increase the probability of reaching the global optimum, it may result in significant losses.

c: COMPUTING TIME (SEC)

Table 4 shows that A-SafeBO, except for GP-UCB*, is the fastest in all environments as it consumes the least time. GP-UCB* is based on ensemble $\mathcal{GP}$ and does not consider safety, so it does not require calculations for the safe set, which makes it faster compared to other safe BO algorithms. A-SafeBO reduces the increase in computation time by using an Ensemble $\mathcal{GP}$ in constructing the surrogate model. All four algorithms used PSO to optimize the acquisition function, so this does not affect the difference in computation time. However, compared to SafeOPT, which requires three swarm optimizers to optimize the acquisition function, A-SafeBO and StageOPT use only one swarm optimizer, thereby having a shorter computation time.

d: RESOURCES USED FOR HYPERPARAMETER TUNING

For SafeOPT and StageOPT, 1100 pre-experiments were performed for hyperparameter tuning in each environment.

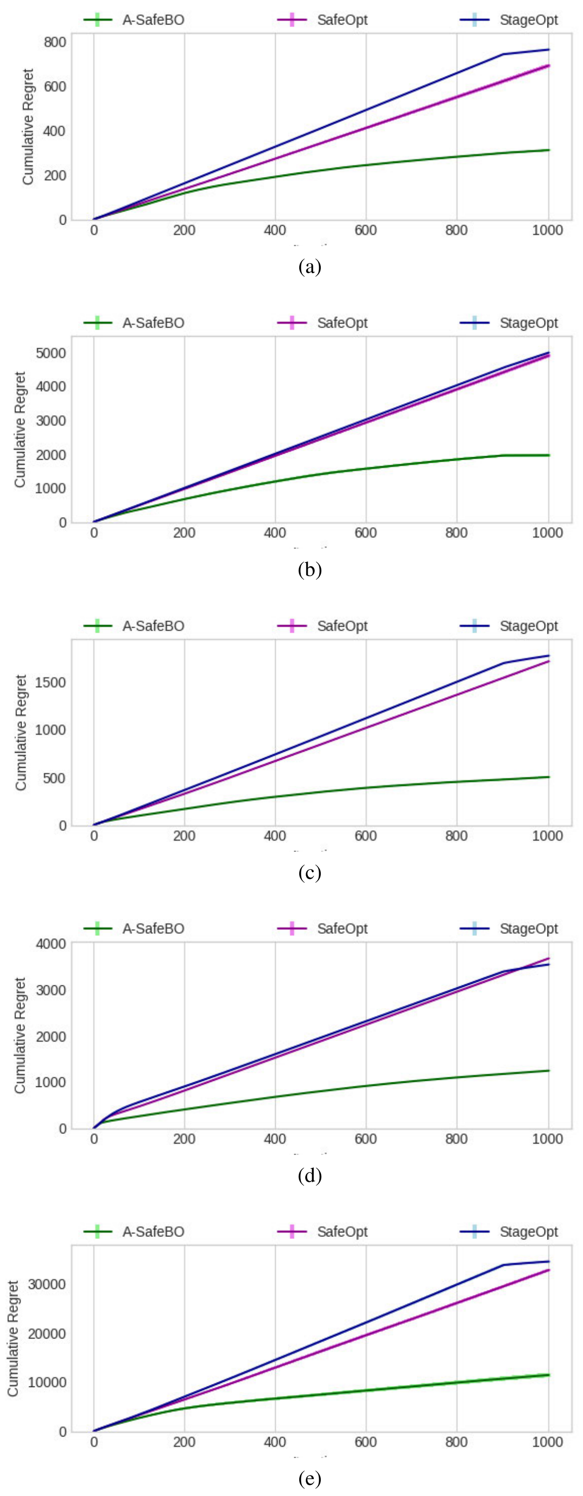

FIGURE 9. Cumulative regrets in each environment (a) Griewank 2-D, (b) Adjiman 2-D, (c) Hartmann 6-D, (d) Periodic 10-D and (e) CCPP when 3 optimization algorithms(SafeOpt, StageOpt and A-SafeBO) are performed using a sufficient number of samples ($T = 1000$).

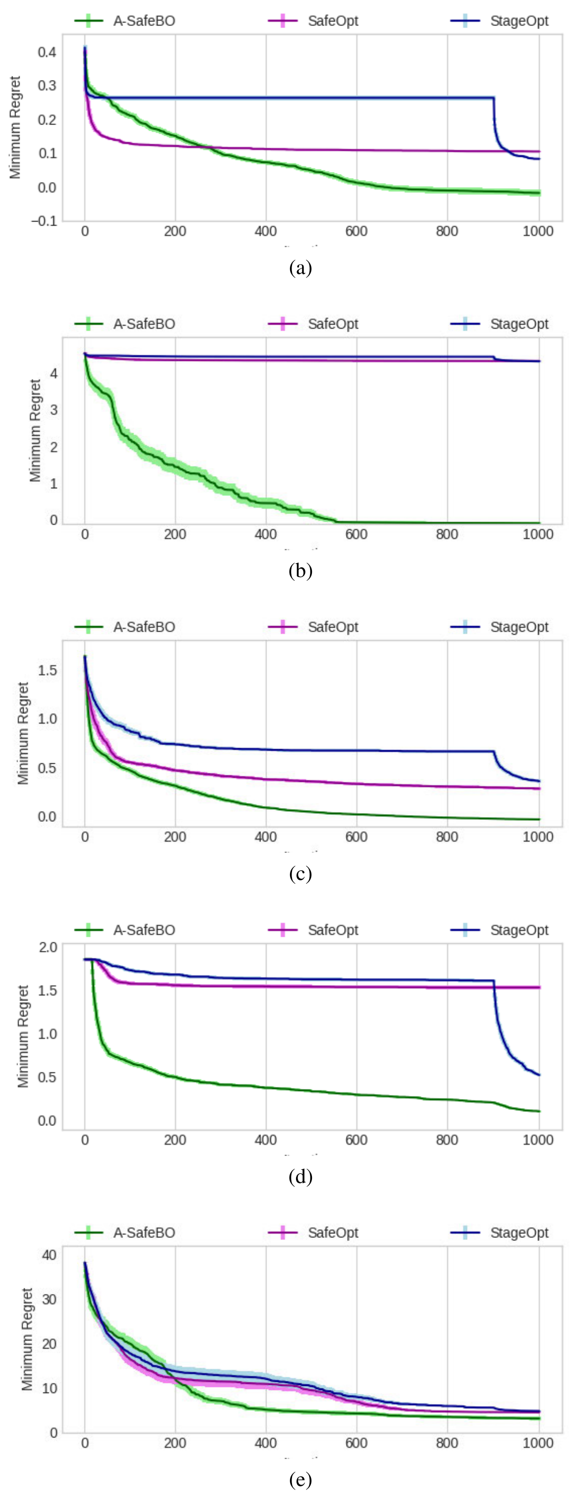

FIGURE 10. Minimum regrets in each environment (a) Griewank 2-D, (b) Adjiman 2-D, (c) Hartmann 6-D, (d) Periodic 10-D and (e) CCPP when 3 optimization algorithms(SafeOpt, StageOpt and A-SafeBO) are performed using a sufficient number of samples ($T = 1000$).

When using SafeOPT and StageOPT, 1,100 times the average of the time consumed in each environment and 550,000 samples were used for hyperparameter tuning. This does not match the initial goal of performing optimization using relatively few samples in a situation where evaluation is expensive, especially in industrial fields.

A-SafeBO can be said to be much more practical in industrial fields where evaluation is expensive because it schedules the hyperparameter scaling process appropriately within a limited number of samples without such prior work.

Taking the experimental results into account, it can be observed that safe BO algorithms exhibit lower performance in terms of speed and the probability of reaching the global optimum, compared to GP-UCB*, due to the consideration of safety. However, it is important to note that these safe BO approaches are crucial in situations where unsafe exploration can lead to significant losses. Additionally, it was confirmed that A-SafeBO outperformed conventional safe BO algorithms in all performance evaluations when a sufficient number of samples were available for the optimization process. A-SafeBO is not only safer and faster than conventional safe BO approaches, but it also explores a wider range, providing a recommendation closer to the global optimum. Furthermore, by eliminating the hyperparameter tuning process, A-SafeBO can save many resources, making it a more practical approach.

2) SMALL NUMBER OF SAMPLES ($T = 100$)

A-SafeBO is a process of obtaining a recommendation with better evaluation through safe exploration in a low evaluation budget environment. To check whether A-SafeBO is an algorithm suitable for these concepts, the performance of A-SafeBO was compared with that of SafeOpt and StageOpt. For comparison, optimization was performed with each algorithm using only a small number of samples ($T = 100$) in each environment. The results of this experiment are shown in Table 5.

a: OVERALL PERFORMANCE COMPARISON

Performance is lower than when optimization is performed using sufficient samples ($T = 1000$) in all environments. This is a natural result since there is not enough information to model the environment. However, when performance is compared based on evaluation metrics under the same conditions, it can be confirmed that A-SafeBO provides a better recommendation quickly without significantly compromising safety considerations, even with a small number of samples in all environments. This is because optimization was performed by effectively controlling exploration and exploitation through the information obtained from the samples without significantly compromising safety through safety prediction $(S^{\varepsilon})_{X^s}$ and acquisition function U proposed by A-SafeBO.

b: PRACTICALITY

As mentioned above, when looking at the experiments using sufficient samples, SafeOpt and StageOpt used many resources (time and samples) to set kernel hyperparameters (length scale l and the bounded RKHS norm B) in each environment in advance for comparison. However, since A-SafeBO does not require this process, many resources consumed for hyperparameter tuning can be saved. These features further increase the practicality of A-SafeBO. This is more meaningful because it is aimed at application in situations where many samples cannot be used.

Through experiments using a small number of samples, it was confirmed that A-SafeBO provides a recommendation with higher evaluation faster than the existing methodologies, even in environments where it is difficult to use a large number of samples for optimization. In addition, it was confirmed that A-SafeBO increases the practicality of the algorithm by saving many resources consumed in tuning the kernel hyperparameters by scheduling the scaling process of kernel hyperparameters appropriately while performing optimization with limited samples through the scaling function h.

VI. CONCLUSION

In this paper, we proposed a practical and safe BO algorithm called A-SafeBO. Our algorithm can be effectively used in real environments that incur high costs in physical resources due to unsafe sampling and are limited by a finite evaluation budget. A-SafeBO proposes new approaches to overcoming the limitations in many conventional approaches.

A-SafeBO uses the Ensemble $\mathcal{GP}$ method to build the surrogate model and uses SafePSO to optimize the acquisition function; therefore, it performs optimization faster than the conventional BO-based methods. In SafePSO, we propose a new acquisition function U, safety prediction method $(S^{\varepsilon})_{X^s}$ and the way to choose the initial set of particles X^s to safely explore a wider range compared to methods that consider existing safety constraints. Through this, A-SafeBO can explore a wider range without significantly compromising the consideration of safety constraints and make recommendations with better performance on average by reducing the influence of the initial point on optimization. Finally, by scheduling the scaling process of kernel hyperparameters using scaling function h within a limited number of samples, the optimization process is performed effectively without the hyperparameter tuning process, saving resources consumed for hyperparameter tuning and presenting a much more practical solution than the conventional methods.

A-SafeBO can be a good solution in fields that require safety considerations (e.g., industrial fields, medical fields, etc.) because evaluation is expensive and unsafe evaluation can cause great losses. In particular, data is collected when there is insufficient data, and optimization can be performed effectively using a relatively small amount of data.

APPENDIX A THEORETICAL PROOF

We assume that the target function f that is corrupted by σ-sub-Gaussian noise is $f : D \rightarrow \mathbb{R}, D \rightarrow \mathbb{R}^d$. The complexity of f is measured by the norm in Reproducing Kernel Hilbert space(RKHS, [17]). RKHS H_θ contain $f(x)$ having the same form $f(x)=\sum_{i\geq 0} \alpha_i k_\theta(x, x_i)$, $\alpha_i \in \mathbb{R}$. α_i is weight that decays sufficiently fast enough and $k_\theta(x, x_i)$ is a kernel that is parameterized by length scale θ.

Lemma 1: We assume that the target function f has the RKHS norm bound like $\|f\|_{H_\theta} \leq B$. In this case, if $\beta_t^{1/2} = B + 4\sigma\sqrt{I(y_t; f) + 1 + \ln(1/\delta)}$, $|f(x) - \mu_t(x)| \leq \beta_t^{1/2}\sigma_t(x)$ is satisfied with at least $1 - \delta$ probability.

$I(y_t; f)$ is the mutual information between the GP prior on f and the observation y_t at time t. The definition is the same as $I(y_A; f) = 0.5 \log|I + \sigma^{-2}\mathbf{K}_A|$ for $x \in A$. $\mathbf{K}_A$ is the kernel matrix($[k(x, x')]_{x,x' \in A}$) and $|\cdot|$ is the determenet.

Proof: As shown in Lemma 4 of [25], the proof is the same except that a time-dependent kernel is used in the proofs of [45] and [46]. □

Lemma 1 indicates that when an appropriate $\beta_t^{1/2}$ value is set, the probability that the target function f exists within the confidence bound inferred by the posterior $\mathcal{GP}$ is high.

Lemma 2: Let $f \in H_{\theta_{t_0}}$ and $\|f\|^2_{H_{\theta_{t_0}}} \leq B_{t_0}$. And if B_t and θ_t are determined as $B_t = \frac{B_0}{h(t)}$ and $\theta_t = h(t)^{\frac{1}{d}}\theta_0$ using monotonically decreasing $0 < h(t) \leq 1$, then for all $t \geq t_0$, $f \in H_{\theta_t}$ and $\|f\|^2_{H_{\theta_t}} \leq B_t$, $\|f\|^2_{H_{\theta_t}} \leq (\Pi_{i=1}^d \frac{[\theta_{t_0}]_i}{[\theta_t]_i})\|f\|^2_{H_{\theta_{t_0}}}$.

Therefore, monotonically decreasing h yields $H_{\theta_t} \supset H_{\theta_{t_0}}$

Proof: Similar to Lemma 3 of [25],

$$\|f\|^2_{H_{\theta_t}} \leq (\Pi_{i=1}^d \frac{[\theta_{t_0}]_i}{[\theta_t]_i})\|f\|^2_{H_{\theta_{t_0}}}$$
$$\leq \frac{(h(t_0)^{\frac{1}{d}}\theta_0)^d}{(h(t)^{\frac{1}{d}}\theta_0)^d} B_{t_0} = \frac{h(t_0)}{h(t)}\frac{B_0}{h(t_0)} = \frac{B_0}{h(t)} \leq B_t.$$

□

By Lemma 2, it can be confirmed that the function determined by gradually decreasing θ_t and gradually increasing B_t based on the monotonically decreasing $h(t)$ includes the previous one. That is, we consider a larger RKHS as a candidate space for f while simultaneously considering a larger norm ball to determine a more complex function as an estimated function.

Theorem 1: Assume that f parameterized by a stationary kernel $k_\theta(x, x')$ with length scale θ has a bounded RKHS norm as $\|f\|^2_{H_\theta} \leq B$.

If the hyperparameters at time t are calculated like $\theta_t = (h(t))^{\frac{1}{d}}\theta_0$ and $B_t = \frac{B_0}{h(t)}$ based on the initial hyperparameters θ_0 and B_0 and monotonically decreasing function $0 < h(t) \leq 1$, and $\beta^{1/2}$ is calculated as $\beta_t^{1/2} = \frac{B_0}{h(t)} + 4\sigma\sqrt{I_{\theta_t}(y_t; f) + 1 + \ln(\frac{1}{\delta})}$, then the cumulative regret(R_t) has the bounded form as follows with at least $(1 - \delta)$ probability.

$$R_t \leq 2B\max(h^{-1}(\max_i \frac{[\theta_0]_i}{[\theta]_i}), h^{-1}(\frac{B}{B_0}))$$
$$+ \sqrt{C_1 t \beta_t^{1/2} I_{\theta_t}(y_t; f)}(C_1 = \frac{8}{\log(1 + \sigma^{-2})})$$

Proof: The proof is the same except that a decreasing scaling function $h(x)$ is used in the proofs of [25]. □

With the hyperparameter scaling in A-SafeBO following Theorem 1, convergence is proven because the cumulative regret (R_t) has a bounded form and R_t is sublinear over iterations.

Lemma 3: Using the Lipschitz constant L, the safety threshold $\tau \in \mathbb{R}$, and the lower confidence bound $l_t(x) = \mu_t(x) - \beta_t^{1/2}\sigma_t(x)$, the safe set is defined as follows.

$$S_t = \bigcup_{x \in S_{t-1}} \{x' \in D | l_t(x) - L \cdot d(x, x') \geq \tau\}$$

($d(x, x')$: distance between x and x')

Then, if it is $\emptyset \subseteq S_0 \subseteq D$, the Reachable safe set($R_\varepsilon(S) := S \cup \{x \in D | \exists x' \in S, f(x') - \varepsilon - L \cdot d(x, x') \geq \tau\}$, $0 < \varepsilon$) has the following properties.

For any $t \geq 1$

(i) $S_0 \subseteq S_t \subseteq S_{t+1}$

(ii) $S_{t_0} \subseteq S_{t_0+T} \Rightarrow R_\varepsilon(S_{t_0}) \subseteq R_\varepsilon(S_{t_0+T})$, $(0 \leq T)$

(iii) $S_{t_0} \subseteq S_{t_0+T} \Rightarrow \bar{R}_\varepsilon(S_{t_0}) \subseteq \bar{R}_\varepsilon(S_{t_0+T})$, $(0 \leq T)$

Proof: The proof is the same in the proofs of Lemma 2 in [13]. □

REFERENCES

[1] H. Cho, Y. Kim, E. Lee, D. Choi, Y. Lee, and W. Rhee, "Basic enhancement strategies when using Bayesian optimization for hyperparameter tuning of deep neural networks," *IEEE Access*, vol. 8, pp. 52588–52608, 2020.

[2] J. Wu, S. Toscano-Palmerin, P. I. Frazier, and A. G. Wilson, "Practical multi-fidelity Bayesian optimization for hyperparameter tuning," in *Uncertainty in Artificial Intelligence*. Boston, MA, USA: Springer, 2020, pp. 788–798.

[3] F. Archetti and A. Candelieri, *Bayesian Optimization and Data Science*, vol. 849. Cham, Switzerland: Springer, 2019.

[4] P. I. Frazier, "Bayesian optimization," in *Recent Advances in Optimization and Modeling of Contemporary Problems*. New York, NY, USA: Informs, 2018, pp. 255–278.

[5] M. Khosravi, A. Eichler, N. Schmid, R. S. Smith, and P. Heer, "Controller tuning by Bayesian optimization an application to a heat pump," in *Proc. 18th Eur. Control Conf. (ECC)*, Jun. 2019, pp. 1467–1472.

[6] J. Kirschner, M. Mutny, N. Hiller, R. Ischebeck, and A. Krause, "Adaptive and safe Bayesian optimization in high dimensions via one-dimensional subspaces," in *Proc. 36th Int. Conf. Mach. Learn.*, Long Beach, CA, USA, vol. 97, Jun. 2019, pp. 3429–3438.

[7] S. Sano, T. Kadowaki, K. Tsuda, and S. Kimura, "Application of Bayesian optimization for pharmaceutical product development," *J. Pharmaceutical Innov.*, vol. 15, pp. 1–11, Mar. 2019.

[8] M. A. Awal, M. Masud, M. S. Hossain, A. A.-M. Bulbul, S. M. H. Mahmud, and A. K. Bairagi, "A novel Bayesian optimization-based machine learning framework for COVID-19 detection from inpatient facility data," *IEEE Access*, vol. 9, pp. 10263–10281, 2021.

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

[10] J. Mockus, *Bayesian Approach to Global Optimization: Theory and Applications*, vol. 37. Cham, Switzerland: Springer, 2012.

[11] Z. Wang, C. Gehring, P. Kohli, and S. Jegelka, "Batched large-scale Bayesian optimization in high-dimensional spaces," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2018, pp. 745–754.

[12] C. Rasmussen, O. Bousquet, U. Luxburg, and G. RÃ¤tsch, "Gaussian processes in machine learning," in *Advanced Lectures on Machine Learning: ML Summer Schools*. Berlin, Germany: Springer, Sep. 2004, pp. 63–71.

[13] Y. Sui, A. Gotovos, J. Burdick, and A. Krause, "Safe exploration for optimization with Gaussian processes," in *Proc. Int. Conf. Mach. Learn.*, vol. 37, Jun. 2015, pp. 997–1005.

[14] Y. Sui, V. Zhuang, J. Burdick, and Y. Yue, "Stagewise safe Bayesian optimization with Gaussian processes," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4781–4789.

[15] D. Jasrasaria and O. E. Pyzer-Knapp, "Dynamic control of explore/exploit trade-off in Bayesian optimization," in *Proc. Comput. Conf. Intell. Comput.*, vol. 1. Springer, 2019, pp. 1–15.

[16] C. K. Williams and C. E. Rasmussen, *Gaussian Processes for Machine Learning*, vol. 2, Cambridge, U.K.: MIT Press, 2006.

[17] I. Steinwart and A. Christmann, *Support Vector Machines*. Cham, Switzerland: Springer, 2008.

[18] M. O. Ahmed, S. Vaswani, and M. Schmidt, ''Combining Bayesian optimization and Lipschitz optimization,'' *Mach. Learn.*, vol. 109, no. 1, pp. 79–102, Jan. 2020.
[19] R. Lam, K. Willcox, and H. David Wolpert, ''Bayesian optimization with a finite budget: An approximate dynamic programming approach,'' in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, R. Garnett, Ed., Red Hook, NY, USA: Curran Associates, 2016.
[20] P. Auer, ''Using confidence bounds for exploitation-exploration trade-offs,'' *J. Mach. Learn. Res.*, vol. 3, pp. 397–422, Mar. 2002.
[21] H. J. Kushner, ''A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise,'' *J. Basic Eng.*, vol. 86, no. 1, pp. 97–106, Mar. 1964.
[22] D. R. Jones, M. Schonlau, and W. J. Welch, ''Efficient global optimization of expensive black-box functions,'' *J. Global Optim.*, vol. 13, no. 4, pp. 455–492, Dec. 1998.
[23] Z. Wang, F. Hutter, M. Zoghi, D. Matheson, and N. De Feitas, ''Bayesian optimization in a billion dimensions via random embeddings,'' *J. Artif. Intell. Res.*, vol. 55, pp. 361–387, Feb. 2016.
[24] K. Peter Wabersich and M. Toussaint, ''Advancing Bayesian optimization: The mixed-global-local (MGL) kernel and length-scale cool down,'' 2016, *arXiv:1612.03117*.
[25] F. Berkenkamp, P. Angela Schoellig, and A. Krause, ''No-regret Bayesian optimization with unknown hyperparameters,'' *J. Mach. Learn. Res.*, vol. 20, no. 50, pp. 1–24, 2019.
[26] Z. Wang and N. de Freitas, ''Theoretical analysis of Bayesian optimisation with unknown Gaussian process hyper-parameters,'' 2014, *arXiv:1406.7758*.
[27] J. Djolonga, A. Krause, and V. Cevher, ''High-dimensional Gaussian process bandits,'' in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 1–10.
[28] P. Rolland, J. Scarlett, I. Bogunovic, and V. Cevher, ''High-dimensional Bayesian optimization via additive models with overlapping groups,'' in *Proc. Int. Conf. Artif. Intell. Statist.*, 2018, pp. 298–307.
[29] M. Mutnỳ and A. Krause, ''Efficient high dimensional Bayesian optimization with additivity and quadrature Fourier features,'' in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2019, pp. 9005–9016.
[30] R. Moriconi, M. P. Deisenroth, and K. S. Sesh Kumar, ''High-dimensional Bayesian optimization using low-dimensional feature spaces,'' *Mach. Learn.*, vol. 109, pp. 1743–1925, 2020.
[31] K. Kandasamy, J. Schneider, and B. Póczos, ''High dimensional Bayesian optimisation and bandits via additive models,'' in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 295–304.
[32] D. J. Rezende, S. Mohamed, and D. Wierstra, ''Stochastic backpropagation and approximate inference in deep generative models,'' in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1278–1286.
[33] D. P. Kingma and M. Welling, ''Auto-encoding variational Bayes,'' 2013, *arXiv:1312.6114*.
[34] M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. M. Ali Eslami, and Y. Whye Teh, ''Neural processes,'' 2018, *arXiv:1807.01622*.
[35] J. González, J. Longworth, D. C. James, and N. D. Lawrence, ''Bayesian optimization for synthetic gene design,'' 2015, *arXiv:1505.01627*.
[36] M. J. Kusner, B. Paige, and J. M. Hernández-Lobato, ''Grammar variational autoencoder,'' in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1945–1954.
[37] F. Hutter, H. H. Hoos, and K. Leyton-Brown, ''Sequential model-based optimization for general algorithm configuration,'' in *Proc. Int. Conf. Learn. Intell. Optim.* Cham, Switzerland: Springer, 2011, pp. 507–523.
[38] M. McLeod, S. Roberts, and M. A. Osborne, ''Optimization, fast and slow: Optimally switching between local and Bayesian optimization,'' in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 3443–3452.
[39] R. P. R. R. Duivenvoorden, F. Berkenkamp, N. Carion, A. Krause, and P. and A. Schoellig, ''Constrained Bayesian optimization with particle swarms for safe adaptive controller tuning,'' *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 11800–11807, 2017.
[40] R. Eberhart and J. Kennedy, ''Particle swarm optimization,'' in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 4, Dec. 1995, pp. 1942–1948.
[41] B. Lakshminarayanan, D. M. Roy, and Y. W. Teh, ''Mondrian forests for large-scale regression when uncertainty matters,'' in *Artificial Intelligence and Statistics*. Boston, MA, USA: Springer, 2016, pp. 1478–1487.
[42] J. Schreiter, D. Nguyen-Tuong, M. Eberts, B. Bischoff, H. Markert, and M. Toussaint, ''Safe exploration for active learning with Gaussian processes,'' in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*. Cham, Switzerland: Springer, 2015, pp. 133–149.
[43] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. Patwary, M. Prabhat, and R. Adams, ''Scalable Bayesian optimization using deep neural networks,'' in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2171–2180.
[44] J. H. Huggins and J. W. Miller, ''Robust inference and model criticism using bagged posteriors,'' 2019, *arXiv:1912.07104*.
[45] Y. Abbasi-Yadkori, ''Online learning for linearly parametrized control problems,'' Ph.D. thesis, Cambridge Univ., Cambridge, U.K., 2012.
[46] S. R. Chowdhury and A. Gopalan, ''On kernelized multi-armed bandits,'' in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 844–853.
[47] A. R. Al-Roomi. (2022). *Unconstrained Single-Objective Benchmark Functions Repository*. Accessed: Mar. 29. 2022. [Online]. Available: https://www.al-roomi.org/benchmarks/unconstrained
[48] A. Thevenot. (2020). *Optimization and Eye Pleasure: 78 Benchmark Test Functions for Single Objective Optimization*. Accessed: Mar. 29, 2022. [Online]. Available: https://github.com/AxelThevenot/Python_ Benchmark_Test_Optimization_Function_Single_Objective
[49] P. TÃfekci and H. Kaya. (2014). *Combined Cycle Power Plant Data Set—UCI Machine Learning Repository*. Accessed: Mar. 29, 2022. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/combined+cycle+power+plant
[50] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, ''Scikit-learn: Machine learning in Python,'' *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, May 2011.
[51] P. Tüfekci, ''Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods,'' *Int. J. Electr. Power Energy Syst.*, vol. 60, pp. 126–140, Sep. 2014.
[52] E. A. Elfaki and A. H. Ahmed, ''Prediction of electrical output power of combined cycle power plant using regression ann model,'' *J. Power Energy Eng.*, vol. 6, no. 12, p. 17, 2018.
[53] F. Berkenkamp. (2020). *Source Code for Safeopt*. Accessed: Mar. 29, 2022. [Online]. Available: https://github.com/befelix/SafeOpt

GUK HAN received the B.S. degree in electrical engineering from Sogang University, South Korea, in 2016, and the M.S. and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Republic of Korea, in 2018 and 2023, respectively. His current research interest includes Bayesian optimization.

JONGOH JEONG (Graduate Student Member, IEEE) received the B.E. degree in electrical engineering from The Cooper Union for the Advancement of Science and Art, New York, NY, USA, in 2020. He is currently pursuing the master's degree in electrical engineering with the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Republic of Korea. His current research interests include computer vision, continual learning, autonomous driving, and Bayesian optimization applications.

JONG-HWAN KIM (Life Fellow, IEEE) received the Ph.D. degree in electronics engineering from Seoul National University, Seoul, Republic of Korea, in 1987. Since 1988, he has been with the School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Republic of Korea, where he is leading the Robot Intelligence Technology Laboratory as a Professor. He is currently the Director of the KoYoung-KAIST AI Joint Research Center and the Machine Intelligence and Robotics Multi-Sponsored Research Platform. He has authored five books, edited ten books, and around 450 refereed papers in technical journals and conference proceedings. His research interests include intelligence technology, machine intelligence learning, and AI robots. He served as an Associate Editor for the IEEE Transactions on Evolutionary Computation and the *IEEE Computational Intelligence Magazine*.