Bayesian Ascent: A Data-Driven Optimization Scheme for Real-Time Control With Application to Wind Farm Power Maximization

Jinkyoo Park, *Student Member, IEEE*, and Kincho H. Law

***Abstract*— This paper describes a data-driven approach for real-time control of a physical system. Specifically, this paper focuses on the cooperative wind farm control where the objective is to maximize the total wind farm power production by using control actions as an input and measured power as an output. For real time, data-driven wind farm control, it is imperative that the optimization algorithm is able to improve a target wind farm power production by executing as small number of trial actions as possible using the wind farm power monitoring data. To achieve this goal, we develop a Bayesian ascent (BA) algorithm by incorporating into the Bayesian optimization framework a strategy that regulates the search domain, as used in the trust region method. The BA algorithm is composed of two iterative phases, namely, learning and optimization phases. In the learning phase, the BA algorithm approximates the target function using Gaussian process regression to fit the measured input and output of the target system. In the optimization phase, the BA algorithm determines the next sampling point to learn more about the target function (exploration) as well as to improve the target value (exploitation). Specifically, the sampling strategy is designed to ensure that the input is selected within a trust region to improve the target value monotonically by gradually changing the input for a target system. The results from simulation studies using an analytical wind farm power function and experimental studies using scaled wind turbines show that the BA algorithm can achieve an almost monotonic increase in the target value.**

***Index Terms*— Bayesian ascent (BA) algorithm, Bayesian optimization (BO), cooperative control, data-driven control, Gaussian process (GP), wind farm control.**

I. Introduction

In a wind farm, wake generated from upstream wind turbines can significantly lower the power production of the downstream wind turbines due to reduced wind speed inside the wake. The interactions among wind turbines through wakes have been experimentally studied [1]–[4]. Wake interactions and their effects among the wind turbines have motivated researchers to develop cooperative wind farm control strategies to mitigate the wake interference and, thus, to maximize the total wind farm power production. Traditionally, an analytical wind farm power function is constructed and used to formulate the optimal control problem to derive the cooperative control inputs [5]–[8]. However, constructing a wind farm power function is challenging, in that it requires extensive calibrations of the wind farm power function for a specific wind turbine model and a specific wind farm site. As an alternative, data-driven approaches have been proposed to derive the optimum control actions using only wind farm power measurement data. For example, the game theoretic search algorithm [9], [10] and the maximum power-point tracking method have been proposed to determine the optimum control actions using only the wind farm output data [11], [12]. For a data-driven approach to be successful, the algorithm should incrementally improve the target value as rapidly as possible. In this paper, we discuss a data-driven optimization algorithm that is designed to find the optimum input by executing a small number of trial actions and to monotonically improve the target value, so that it can be used for controlling a physical system, such as a wind farm, in real time.

Controlling a physical system using the limited data has been discussed in the context of sequential decision-making problem that seeks to learn the target system (exploration) and find the optimum input for the target system (exploitation) simultaneously. For example, the problem has been discussed in the context of multi-armed bandit problems, in which maximizing the total payoff is compared with gambler's choosing one of a finite number of slot machines with different payoff distributions; the gambler needs to carefully explore different slots machines by playing them and optimally choose the slot machines to play in an attempt to maximize the payoff simultaneously [13]–[15]. Reinforcement learning based on Markov decision processes is also variations of the bandit problems, in that they search the optimum decision/control policy by simultaneously learning and optimizing the objective function (i.e., payoff or reward). Such methods in general require a large number of sampling points to exhaustively explore the input space and, thus, to reach the optimum [16]. By making assumption about the structure of a target function, i.e., representing the target function using some basis functions, the optimum can be more effectively searched. For example, Bayesian optimization (BO) puts prior on a target function using Gaussian process (GP) to describe the overall structure (smoothness) of the target function. The BO algorithm iteratively approximates the input and the output relationship of a target system using GP regression and uses the learned model to determine the inputs that improve the

Manuscript received July 2, 2015; revised October 20, 2015; accepted November 27, 2015. Manuscript received in final form December 4, 2015. Recommended by Associate Editor L. Fagiano.

The authors are with the Civil and Environmental Engineering Department, Stanford University, Stanford, CA 94305 USA (e-mail: jkpark11@stanford.edu; law@stanford.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCST.2015.2508007

target values [17]–[20]. Although exploiting the structure of a target function reduces the search space, the search-based optimization still requires significant samplings before reaching the optimum. The amount of samplings or trials by search-based algorithms hinders their use for real-time control applications.

Derivative-free optimization methods have been developed to optimize an analytical function, whose gradient or Hessian evaluations are not available or expensive to compute. One particular example is the gradient-free trust region method, employed in optimization codes, such as UOBYQA [21], NEWOUA [22], STRONG [23], Booster [24], and ORBIT [25]. The concepts and the performances of various derivative-free optimization methods are compared in [26]. These methods sequentially construct a local function, i.e., a quadratic function [21]–[23] or a radial basis function [24], [25], to approximate the target function using the sampled inputs and the corresponding function values and optimize the approximated local function (surrogate model) to determine the next input. In optimizing the local surrogate model, a trust region is imposed in the scope of the next available solution to make sure that the approximation is close enough to the true function and the next input improves the target function value. Although gradient-free methods require a large amount of input and output data to fit a local surrogate model to the target function at every iteration, the trust region concept helps achieve a monotonic increase in the target value, which is a desirable characteristic in optimizing a physical system.

In this paper, we develop what we call a Bayesian ascent (BA) algorithm that combines the strengths of the BO and gradient-free trust region algorithms. The BA algorithm is built upon the BO framework, so that the target function can be efficiently modeled using GP with a small number of data points. Furthermore, the algorithm incorporates into a BO framework a strategy that regulates the search domain, as used in the trust region method. Due to the use of the trust region constraint in the sampling procedure, BA tends to increase the target value monotonically by gradually changing the inputs of the target system. The monotonic increase in the target value is especially beneficial when applying BA to real-time control applications because the sampling inputs that lead to inferior system responses can be avoided. In addition, the gradual change in the input is a desirable characteristic for controlling a physical system, since the control actions of a physical system are often difficult to change abruptly.

The BA algorithm is tested to determine the optimum coordinated control actions that maximize the power production of a wind farm. First, we assess the performance of the BA algorithm in optimizing the analytical wind farm power function derived in [8]. The performance of the BA algorithm is measured in terms of convergence rate toward the theoretical maximum determined by employing mathematical programming to the target wind farm power function. Furthermore, to demonstrate the applicability of the BA algorithm as a model-free, data-driven approach for real-time control, experimental studies using scaled wind turbines are conducted. In this scenario, an analytical wind farm power function is not available. The results illustrate the potential and effectiveness of the BA algorithm for real-time control of a physical system.

This paper is organized as follows. First, BO is briefly introduced to provide the context for the BA algorithm. Then, the BA algorithm is described and its characteristics are discussed by applying the algorithm for optimizing a selection of 2-D functions. The proposed BA algorithm is then applied to a set of test functions with different dimensions. Finally, the BA algorithm is employed to the cooperative wind farm control problem that seeks to find the optimum coordinated control actions for wind turbines using the measured input and output data from the wind turbines in a wind farm. Both the simulation and experimental studies are discussed. This paper is concluded with a brief summary and discussion.

II. Bayesian Optimization

BO seeks to solve $\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}} f(\boldsymbol{x})$ by iteratively choosing the input $\boldsymbol{x} = (x_1, \ldots, x_i, \ldots x_m)$, where m denotes the dimension of input $\boldsymbol{x}$, and observing the corresponding noisy response $y = f(\boldsymbol{x}) + \epsilon$, where ϵ represents the noise that is assumed to follow a Gaussian distribution, i.e., $\epsilon \sim N(0, \sigma_\epsilon^2)$ [17]–[19]. The function $f(\boldsymbol{x})$, whose analytical expression is unknown, represents the model of the target system. To construct the regression model for the unknown target function $f(\boldsymbol{x})$ and maximize it at the same time, each iteration of BO consists of two phases, namely, learning and optimization.

A. Learning Phase

For the nth iteration in the learning phase of BO, using the selected inputs $\boldsymbol{x}^{1:n} = \{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n\}$ and the corresponding observed outputs $\boldsymbol{y}^{1:n} = \{y^1, \ldots, y^n\}$, the unknown objective function $f(\boldsymbol{x})$ is modeled using a GP regression. In GP regression, the prior on the function values $\boldsymbol{f}^{1:n} = \{f^1, \ldots, f^n\}$, where $f^i = f(\boldsymbol{x}^i)$, is represented as a GP, i.e., $p(\boldsymbol{f}^{1:n}) = \mathrm{GP}(m(\cdot), k(\cdot,\cdot))$, where $m(\cdot)$ is a mean function and $k(\cdot,\cdot)$ is a kernel function. The mean function $m(\boldsymbol{x})$ describes the overall trend in the function values, which are represented by various basis functions, such as linear or periodic functions. If there is no prior knowledge about the target function $f(\boldsymbol{x})$, setting $m(\boldsymbol{x}) = \boldsymbol{0}$ greatly simplifies the learning and prediction procedures of GP without losing the representability of the GP model. The kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$ quantifies the geometrical similarity between two inputs, $\boldsymbol{x}$ and $\boldsymbol{x}'$, which is used to approximate the covariance in their function values. In addition, the likelihood function of the measured response $\boldsymbol{y}^{1:n}$ is represented as a Gaussian distribution, i.e., $p(\boldsymbol{y}^{1:n}|\boldsymbol{f}^{1:n}) = N(\boldsymbol{f}^{1:n}, \sigma_\epsilon^2 \mathbf{I})$. Given the prior and the likelihood assumed, the function value $f(\boldsymbol{x})$, denoted here by f, for an unseen input $\boldsymbol{x}$, and the observed outputs $\boldsymbol{y}^{1:n} = \{y^1, \ldots, y^n\}$ follow a multivariate Gaussian distribution [27]:

$$\begin{bmatrix} \boldsymbol{y}^{1:n} \\ f \end{bmatrix} \sim N\left(\boldsymbol{0}, \begin{bmatrix} (\mathbf{K} + \sigma_\epsilon^2 \mathbf{I}) & \boldsymbol{k} \\ \boldsymbol{k}^T & k(\boldsymbol{x}, \boldsymbol{x}) \end{bmatrix}\right) \tag{1}$$

where $\boldsymbol{k}^T = (k(\boldsymbol{x}^1, \boldsymbol{x}), \ldots, k(\boldsymbol{x}^n, \boldsymbol{x}))$ and $\mathbf{K}$ is the covariance matrix (kernel matrix) whose (i, j)th entry is $\mathbf{K}_{ij} = k(\boldsymbol{x}^i, \boldsymbol{x}^j)$.

The noise variance σ_ϵ^2 quantifies the level of noise that exists in the noisy response $y = f(\boldsymbol{x}) + \epsilon$.

The type of kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$ used to build a GP regression model and its parameters strongly affects the overall representability of the GP regression model. We use a squared exponential covariance function whose evaluation between two input vectors $\boldsymbol{x}^i$ and $\boldsymbol{x}^j$ is expressed as [28]

$$k(\boldsymbol{x}^i, \boldsymbol{x}^j) = \sigma_s^2 \exp\left(-\frac{1}{2}(\boldsymbol{x}^i - \boldsymbol{x}^j)^T \mathrm{diag}(\boldsymbol{\lambda})^{-2}(\boldsymbol{x}^i - \boldsymbol{x}^j) \right) \quad (2)$$

which is described by parameters, σ_s and $\boldsymbol{\lambda}$. The term σ_s^2 is referred to as the signal variance that quantifies the overall magnitude of the covariance value. The parameter vectors $\boldsymbol{\lambda} = (\lambda_1, \ldots \lambda_i, \ldots, \lambda_m)$ is referred to as the characteristic length scales to quantify the relevancy of the input features in $\boldsymbol{x} = (x_1, \ldots x_i, \ldots, x_m)$ for predicting the response y. A large length scale λ_i indicates weak relevance, while a small length scale λ_i implies the strong relevance of the corresponding input feature x_i.

The hyperparameters $\boldsymbol{\theta} = (\sigma_\epsilon, \sigma_s, \boldsymbol{\lambda})$ for the noise model and the kernel function are determined as ones maximizing the marginal log-likelihood of the training data $\boldsymbol{D}^n = \{(\boldsymbol{x}^i, y^i) | i = 1, \ldots, n\}$ as [27]

$$\begin{aligned} \boldsymbol{\theta}^* &= \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{y}^{1:n} | \boldsymbol{x}^{1:n}, \boldsymbol{\theta}) \\ &= \arg\max_{\boldsymbol{\theta}} \left(-\frac{1}{2}(\boldsymbol{y}^{1:n})^T \left(\mathbf{K} + \sigma_\epsilon^2 \mathbf{I}\right)^{-1} \boldsymbol{y}^{1:n} \right. \\ &\quad \left. - \frac{1}{2}\log\left|\mathbf{K} + \sigma_\epsilon^2 \mathbf{I}\right| - \frac{n}{2}\log 2\pi \right). \end{aligned} \quad (3)$$

With the measurement data and the hyperparameters updated (i.e., optimized), the posterior distribution on the response f for the unseen input $\boldsymbol{x}$ given the historical data $\boldsymbol{D}^n = \{(\boldsymbol{x}^i, y^i) | i = 1, \ldots, n\}$ can be expressed as an 1-D Gaussian distribution $f \sim N(\mu(\boldsymbol{x}|\boldsymbol{D}^n), \sigma^2(\boldsymbol{x}|\boldsymbol{D}^n))$ with the mean and variance functions expressed, respectively, as [27]

$$\mu(\boldsymbol{x}|\boldsymbol{D}^n) = \boldsymbol{k}^T (\mathbf{K} + \sigma_\epsilon^2 \mathbf{I})^{-1} \boldsymbol{y}^{1:n} \quad (4)$$

$$\sigma^2(\boldsymbol{x}|\boldsymbol{D}^n) = k(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{k}^T (\mathbf{K} + \sigma_\epsilon^2 \mathbf{I})^{-1} \boldsymbol{k}. \quad (5)$$

Here, $\mu(\boldsymbol{x}|\boldsymbol{D}^n)$ and $\sigma^2(\boldsymbol{x}|\boldsymbol{D}^n)$ are used as the functions for evaluating, respectively, the mean and the variance of the hidden function output f corresponding to the unseen input data $\boldsymbol{x}$.

B. Optimization Phase

For the nth iteration in the optimization phase of BO, the GP mean function $\mu(\boldsymbol{x}|\boldsymbol{D}^n)$ and the variance function $\sigma^2(\boldsymbol{x}|\boldsymbol{D}^n)$, which describe the distribution of the function value $f(=f(\boldsymbol{x}))$ at an unobserved input $\boldsymbol{x}$ at the nth iteration, can be used to select the next input $\boldsymbol{x}^{n+1}$ in order to learn more about the target function as well as to improve the target value at the same time. If we are to learn about the target system only, one possible strategy would be to select the next input as the one that maximizes the variance function $\sigma^2(\boldsymbol{x}|\boldsymbol{D}^n)$, which reduces the uncertainty (variance) around the selected input (exploration). This sampling strategy, called active learning, has been widely used to select sampling points or to determine experimental parameters [29]. On the other hand, if we are to increase the target value, the natural strategy would be to select the next input as the one maximizing the mean function $\mu(\boldsymbol{x}|\boldsymbol{D}^n)$ that reflects the current belief about the target system (exploitation). However, this strategy tends to be too greedy attempting to search for a superior target value than the current one. For the success of BO, the balance between learning the target function (exploration) and maximizing the target function (exploitation) is required.

In general, the next sampling point is being selected as one that maximizes the acquisition function that incorporates both aspects of exploration and exploitation. For example, Cox and John [30] proposed a method of selecting the next input as the one that maximizes the upper confidence bound (UCB) acquisition function as

$$\boldsymbol{x}^{n+1} = \arg\max_{\boldsymbol{x}} \left(\mu(\boldsymbol{x}|\boldsymbol{D}^n) + \rho^n \sigma(\boldsymbol{x}|\boldsymbol{D}^n) \right) \quad (6)$$

where the parameter ρ^n is selected to balance between the exploration and the exploitation. If ρ^n is small, $\boldsymbol{x}^{n+1}$ is selected as the one that maximizes the mean $\mu(\boldsymbol{x}|\boldsymbol{D}^n)$ (exploiting). On the other hand, if ρ^n is large, $\boldsymbol{x}^{n+1}$ is selected as one that is associated with a large variance (exploring). In addition, Mockus *et al.* [31] proposed a method of selecting the next input based on maximizing the expected improvement (EI) acquisition function as

$$\boldsymbol{x}^{n+1} = \arg\max_{\boldsymbol{x}} \mathrm{E}[\max\{0, f - f^{\max}\} | \boldsymbol{D}^n] \quad (7)$$

where $\max\{0, f - f^{\max}\}$ is the improvement toward the maximum output f compared with the maximum target function value $f^{\max} = \max_{\{\boldsymbol{x} \in \boldsymbol{x}^{1:n}\}} \mu(\boldsymbol{x}|\boldsymbol{D}^n)$ that is estimated in the (current) nth iteration. Note that the improvement is quantified with respect to the estimated maximum function value $f^{\max}$ rather than the actually observed maximum response $y^{\max}$. This is because the measurement value $y^{\max}$ may have large noise and may, thus, interfere the sampling procedure. Given the estimated $f^{\max}$, the value of the $\mathrm{EI}(\boldsymbol{x})$ can be analytically derived using the distribution of the target function f at $\boldsymbol{x}$ [31]. The EI function $\mathrm{EI}(\boldsymbol{x})$ has a higher value when either the mean or the variance is large. Therefore, by selecting $\boldsymbol{x}$ that maximizes $\mathrm{EI}(\boldsymbol{x})$, we can obtain either the improved objective function value (exploitation) or the updated objective function with reduced uncertainty (exploration) at that point.

III. Bayesian Ascent Algorithm

The BO algorithm can effectively optimize a target function using a limited number of sampled data points from a target system. However, the BO algorithm with commonly used acquisition functions, such as EI and UCB, tends to select the inputs over a large input space when exploring the target system. Two issues arise when the BO algorithm is used to control a physical system: 1) the difference between the successive inputs is often too large and 2) the inputs are chosen from the region where the uncertainty is too large. Often, control actions, i.e., the inputs, cannot be changed abruptly in a physical system and the selected action chosen from a highly uncertain input space can result in significantly inferior target value. So far, the BO algorithm has not been widely applied to real-time control applications. In this section, we discuss a modified sampling strategy for the BO algorithm so that the algorithm can be efficiently applied to real-time control

applications. Specifically, we impose a proximity constraint in solving (7), such that the BO algorithm would sample target values that are near the best solution observed so far. This strategy is similar to imposing a trust region constraint in mathematical programming. We also strategically adjust the size of the trust region to expedite the rate of convergence to an optimum.

A. Hypercube Trust Region

Gradient-based mathematical optimization searches for a (local) optimum by iteratively optimizing a model function, which is in general a quadratic function, constructed using the gradient and the Hessian at each iteration. To guarantee that the next solution is selected from the region where the approximated model function is close to the target function, a trust region is defined to impose a proximity constraint, such that the next iterate is selected only within the trust region [32]. The size of the trust region can be adjusted depending on the improvement in the target value to guarantee that the algorithm converges to the (local) optimum. Specifically, if the observed increase with respect to the previous function value is larger than a certain threshold of the predicted increase from the approximate model function, the solution will be used as the next iteration, and the trust region is expanded to expedite the convergence rate. Otherwise, the current solution will be rejected, and the trust region will be contracted to find a solution nearer to the current solution. Because of this strategic adjustment in the trust region method, a target value would tend to increase monotonically, and the solution at each iteration converges gradually toward a (local) optimum.

To take advantages (i.e., the monotonic increase in a target value and the gradual convergence to an optimum) of the trust region method, we impose the trust region constraint to the optimization phase of BO. With a trust region constraint imposed, BO then finds the next input close to the input $\boldsymbol{x}^{\max}$ that corresponds to the best target value $f^{\max}$ observed so far. That is, BO selects the next input from the region where the uncertainty of the model function is not large (i.e., the GP mean function is close to the underlying target function around the region with sampled points). The optimization phase of BO can be posed as a constrained optimization problem described as

$$\begin{aligned} &\max_x \mathrm{E}[\max\{0, f - f^{\max}\}|\boldsymbol{D}^n] \\ &\text{s.t. } \mathbf{x} \in \mathbf{T} := \left\{\boldsymbol{x} \middle| \left\|x_i - x_i^{\max}\right\| < \tau_i \text{ for } i = 1, \ldots, m\right\} \end{aligned} \quad (8)$$

where the objective function is the EI, which is given in (7). The trust region $\boldsymbol{T}$ is defined as a hypercube with its center being $\boldsymbol{x}^{\max}$; the ith component τ_i of $\boldsymbol{\tau} = (\tau_1, \ldots \tau_i, \ldots, \tau_m)$ determines the range where the ith feature x_i of $\boldsymbol{x} = (x_1, \ldots x_i, \ldots, x_m)$ to be sampled next. Thus, the vector $\boldsymbol{\tau}$ controls the overall size of the hypercube trust region where the exploration takes place.

We call the BO algorithm with the trust region constraint the Bayesian ascent (BA) algorithm, in that the algorithm follows the ascending direction estimated probabilistically from the sequence of observations. The overall procedure is summarized in Algorithm 1. As in the trust region method,

Algorithm 1 Bayesian Ascent (BA) Algorithm

Choose $\boldsymbol{x}^1$ and $\boldsymbol{\tau}^1$ (initial trust region size) and observe y^1
Repeat until convergence, $n = 1, 2, 3 \ldots$
1) Optimize the hyper parameters
$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \log p\,(\boldsymbol{y}^{1:n}|\boldsymbol{x}^{1:n}, \boldsymbol{\theta})$
2) Construct a GP model
$\mu(\boldsymbol{x}|\boldsymbol{D}^n) = \boldsymbol{k}^T(\mathbf{K} + \sigma_\epsilon^2\mathbf{I})^{-1}\boldsymbol{y}^{1:n}$
$\sigma^2(\boldsymbol{x}|\boldsymbol{D}^n) = k(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{k}^T(\mathbf{K} + \sigma_\epsilon^2\mathbf{I})^{-1}\boldsymbol{k}$
3) Select the next input by solving
$\boldsymbol{x}^{n+1} = \arg\max_{\boldsymbol{x}\in\boldsymbol{T}} \mathrm{E}[\max\{0, f - f^{max}\}|\boldsymbol{D}^n]$
s.t. $\boldsymbol{x} \in \boldsymbol{T} := \{\boldsymbol{x}| \, \|x_i - x_i^{max}\| < \tau_i^n \text{ for } i = 1, \ldots, m\}$
4) Append the data $\boldsymbol{D}^{n+1} = \left\{\left(\boldsymbol{x}^i, y^i\right)| \, i = 1, \ldots, n+1\right\}$
5) Update size of trust region
if $y^{n+1} - f^{max} \geq \gamma(1/n)(f^{max} - y^1)$ **then**
$\boldsymbol{\tau}^{n+1} = \beta\boldsymbol{\tau}^n, (\beta > 1)$
else
$\boldsymbol{\tau}^{n+1} = \boldsymbol{\tau}^1$
end if

BA adjusts the size of the trust region depending on the improvement in the target value. Let us denote the solution of (8) as $\boldsymbol{x}^{n+1}$. We check whether $\boldsymbol{x}^{n+1}$ sufficiently improves the target value. With the measured output response y^{n+1}, if the observed increase $y^{n+1} - f^{\max}$ with respect to the previously estimated value $f^{\max}$ is larger than a certain threshold ratio γ of the average increase $(1/n)(f^{\max} - y^1)$, where y^1 is the initial measurement, the input $\boldsymbol{x}^{n+1}$ will be updated as the best solution $\boldsymbol{x}^{\max}$ observed so far, and the trust region is expanded as $\boldsymbol{\tau}^{n+1} = \beta\boldsymbol{\tau}^n$, with $\beta > 1$ to expedite the convergence rate. Otherwise, the trust region will be reset as $\boldsymbol{\tau}^{n+1} = \boldsymbol{\tau}^1$, where $\boldsymbol{\tau}^1$ is the size of initial trust region. Here, we use the average increase $(1/n)(f^{\max} - y^1)$ as the criterion for evaluating the improvement as it represents a more reliable criterion than a point value that is susceptible to randomness. The BA algorithm uses the entire historical input and output data to construct the surrogate model function to approximate the target function, which is different from the gradient-free trust region method that uses only a subset of local data points to construct a local (quadratic) model function. In addition, the BA algorithm does not contract the size of the trust region $\boldsymbol{\tau}$ but resets it to the initial size $\boldsymbol{\tau}^1$ because continuously contracting the trust region can possibly cause the sampling to be staying at an arbitrary point that is not a (local) optimum. When a (local) optimum is trapped by the hypercube, BO cannot find the input that results in a target value larger than the local optimum inside the trust region, leading the BA algorithm to converge to the (local) optimum. Note that this paper uses $\gamma = 0.05$ and $\beta = 1.1$. In addition, the initial trust region $\boldsymbol{\tau}^1$ can be determined considering the ranges of the input component. Using 1%∼10% of the input range for each input component works well in general. If the initial trust region is small, in general, a large number of iterations are required for the BA algorithm to reach an optimum. On the contrary, if the initial trust region is large, the convergence rate can be expedited, but the trend of monotonic

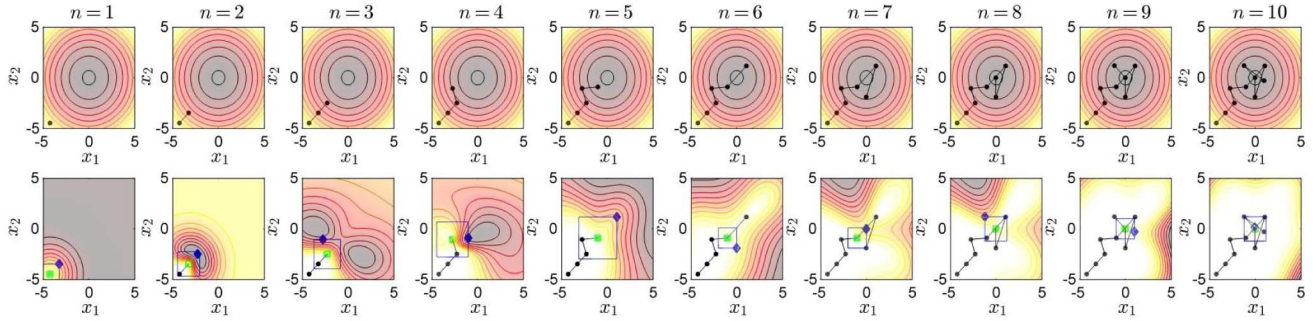

Fig. 1. BA method for finding the optimum of a 2-D quadratic function.

increase in the target value can be weakened. Note that when the initial trust region is set to be the same size of the input space, the BA algorithm becomes equivalent to the BO algorithm.

B. Illustrative Examples

1) 2-D Quadratic Function: This example illustrates the basic step of the BA algorithm. Fig. 1 shows how the BA algorithm finds the optimum of a 2-D quadratic function $f(\boldsymbol{x}) = -(x_1^2 + x_2^2)/50 + 1$. The figures in the upper row show the trajectory of sampled inputs overlaying on the true function represented as contours. The figures in the bottom row show how the next sampling point is being determined at each iteration. In each figure, the circular dots are the inputs that have been sampled so far and the field square is the best input among the sampled inputs in the current iteration. Based on the sampled inputs and the corresponding target values, the EI function, shown as contour plots in the bottom figures, is constructed. The next sampling point is then determined by maximizing the EI function within the trust region, as shown by the open square. The next selected input is marked as field diamond shown in the figures. For this example, we use the initial trust region that has a size of 10% of the input range in each component, i.e., $\tau_i^1 = 0.1 \times |\max(x_i) - \min(x_i)|$ for $i = 1, \ldots, m$.

As shown in Fig. 1, for the four iterations ($n = 1, \ldots, 4$), the target value increases monotonically and the trust region expands by setting $\boldsymbol{\tau}^{n+1} = \beta \boldsymbol{\tau}^n$ with $\beta = 1.1$. During the fifth iteration ($n = 5$), it is observed that the newly sampled value does not increase the target value, i.e., the value does not satisfy the condition of $y^6 - f^{\max} > \gamma\,(1/5)(f^{\max} - y^1)$. As a result, the size of the trust region resets to $\boldsymbol{\tau}^6 = \boldsymbol{\tau}^1$ and the center $\boldsymbol{x}^{\max}$ of the trust region remains unchanged. The same situation occurs at $n = 6$. With the learned information from these two failed steps, the iteration proceeds to sample the input that increases the target value. Due to the use of the trust region constraint, the BA algorithm improves the target value monotonically except during the iterations where BA tries to find the ascending direction ($n = 5$ and 6). In addition, the BA algorithm regulates the step size, i.e., the distance between two successive samples, which can be beneficial in controlling a physical system. This is because, for a physical system, changing the control inputs gradually (progressively) is more preferable than changing the control inputs radically.

2) 2-D Function With Two Local Optima: With the trust region constraint, the BA algorithm is able to increase a target value monotonically by gradually changing the input values. As a result, the BA algorithm seeks to find a local optimum, similar to gradient-free trust region optimization algorithms. In this section, we investigate the local convergence characteristics of BA using a simple analytical function. Let us consider the function $f(\boldsymbol{x}) = \exp(-((x_1 - 3)^2/22) - ((x_2 - 2.5)^2/16)) + 0.6\exp(-((x_1 + 3.5)^2/12) - ((x_2 + 2.5)^2/16))$ which has two local optima. In this example, for each simulation, we randomly sample an initial point from one of the two regions $R_1 = \{\boldsymbol{x} | -5 < x_1 < -1, -5 < x_2 < 0$ and $R_2 = \{\boldsymbol{x} | -1 < x_1 < 5, 0 < x_2 < 5$, each has its local optimum. We then track the iterations to see how the target value sampled by the BA algorithm approaches to the local optimum for the region where the initial point resides. Furthermore, to test whether the BA algorithm can identify and follow the ascending direction even with noisy data, we use noisy target value $y^n = f(\boldsymbol{x}^n) + \epsilon$, with Gaussian error $\epsilon \sim N(0, \sigma_\epsilon^2)$ with $\sigma_\epsilon = 0.01$ (1%).

Fig. 2 summarizes the results depicting how the BA algorithm converges to a local optimum based on an initial point. Fig. 2(a) shows the results of 1000 simulations using 1000 initial points randomly sampled from region R1. Starting at an initial point ($n = 1$), the BA algorithm proceeds to sample the target values, shown as dots in each figure, until it reaches the local optimum in the region nearest to the initial point. Each dot in the subfigure ($n = 2$ to $n = 19$) represents the sampled inputs during an optimization step. Fig. 2(a) shows that the population of the sampled points converges to the (local) optimal region $\{\boldsymbol{x} | f(\boldsymbol{x}) \geq f(\boldsymbol{x}^*) - \sigma_{\text{error}}\}$ within about 15 iterations (i.e., using 15 evaluated function values). Note that the BA algorithm does not necessarily locate the exact local optimum because of the noise added to the target value. Fig. 2(b) shows the result for the same simulations using the 1000 initial points sampled from region R2. The simulation studies suggest that the BA algorithm can effectively locate a (local) optimum using the strategic sampling strategy based on the trust region method. In particular, the rapid convergence to a (local) optimum can be useful when the BA algorithm is applied to the real-time control application. That is, given any initial point, the BA algorithm is able to find an input that provides a target value that is superior to that of the initial point.

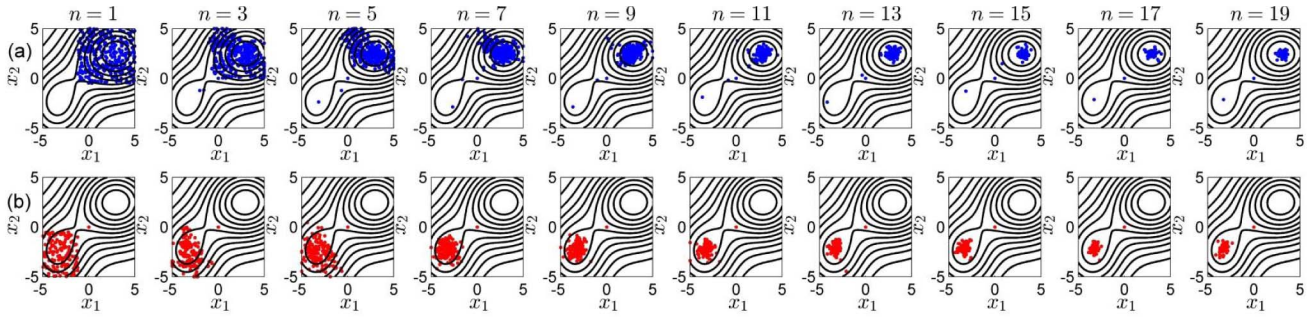

Fig. 2. Convergence to local optima for a 2-D function with two peaks: (a) shows the convergence trend with the initial points sampled from $R_1 = \{x | -5 < x_1 < -1, -5 < x_2 < 0\}$ and (b) shows the convergence trend with the initial points sampled from $R_2 = \{x | -1 < x_1 < 5, 0 < x_2 < 5\}$.

TABLE I
LIST OF TEST FUNCTIONS

Function	n	Hypercube bounds
Wood	4	$[-3,2]^4$
Trigonometric	5	$[-1,2]^5$
Discrete Boundary Value	8	$[-3,3]^8$
Extended Rosenbrock	10	$[-2,2]^{10}$
Variably Dimensioned	10	$[-2,2]^{10}$
Broyden Tridiagonal	11	$[-1,1]^{11}$
Extended Powell Singular	16	$[-1,3]^{16}$

IV. APPLICATION TO BENCHMARKING PROBLEMS

We evaluate the performance of the BA algorithm using a set of test functions that have been designed and used previously to evaluate the performance of derivative-free optimization algorithms. The BA algorithm is employed to find the optimum value of a test function by sequentially sampling its function values. The main goal of this simulation study is to investigate how fast the BA algorithm approaches the optimum values of the target functions with different dimensions. By using different test functions with a broad range of input dimensions, we can test the performance of the BA algorithm for its potential applications to various physical systems.

A. Test Functions

We employ the same set of test functions used in [25] for testing a derivative-free trust region optimization algorithm, referred to as ORBIT. The test functions are a subset of the More–Garbow–Hillstrom test function collection for unconstrained optimization [32]. The test functions and their dimensions and optimization scopes, i.e., boundary constraints, are shown in Table I. These test functions are known to have a single local optimum (global optimum) whose value is 0.

The functions in Table I are used to evaluate the performance of the BA algorithm. For each function, we execute 100 times of BA optimization. For each simulation run, we randomly choose an initial solution inside the hypercube bounds. In addition, at each iterate, noise is added to the sampled function value as $f(\boldsymbol{x})(1+\epsilon)$, i.e., the sampled value is the sum of the function value $f(\boldsymbol{x})$ and the noise term $f(\boldsymbol{x})\epsilon$. The error term ϵ for scaling the noise term is assumed to follow the Gaussian distribution, i.e., $\epsilon \sim N(0, \sigma_\epsilon^2)$. By setting the level of noise σ_ϵ^2, this approach allows adding similar level of noise terms to a target function regardless of the output scale of the target function. The effect of the noise variance σ_ϵ^2 on the effectiveness of the BA algorithm is also studied. Note that the test functions in Table I were designed to test the gradient-free optimization algorithms that are devised to minimize these functions. Here, we maximize the negative of these functions, which is the same as the minimization of the functions.

B. Results

Fig. 3 shows the improvements in the values of the seven target functions with the iterations of the BA algorithm. Each plot shows 100 trajectories of the sampled function values by the BA algorithm, each of which starts from a randomly selected initial solution. For the simulations, $\sigma_\epsilon = 0.01$ is used for the standard deviation of the Gaussian noise ϵ that is used to compute the noisy output $y = f(\boldsymbol{x})(1+\epsilon)$ from the target function $f(\boldsymbol{x})$. As shown in Fig. 3, for each test function, the outputs for the randomly chosen 100 initial inputs disperse a lot. As the BA algorithm proceeds, the function values sampled by the BA algorithm monotonically increase, reaching convergence after 50 iterations for most of the test functions. Although the convergence rates vary for the seven test functions, the required numbers of function evaluations for the BA algorithm to converge are remarkably small even for a high-dimensional function (with optimization variables more than 10).

The performance of an optimization algorithm is often evaluated in terms of the percentage increase in the function value with respect to the initial function value that the algorithm starts with [25]. Using the same simulation results, we quantify the improvement in a target response by each run of the BA algorithm. That is, we normalize the distance from the initially sampled output and the analytical maximum to be 1. Then, the improvement in the output by each iteration of the BA algorithm is represented as an improvement ratio. For example, the improvement ratio of 0 means the initial solution, while the improvement ratio of 1 means the BA algorithm reaches the analytical maximum. The 100 output trajectories for each function in Fig. 3 are converted into the trajectories of the improvement ratio. Averaging 100 trajectories of the improvement ratio for each test function, the average improvement ratio curve is obtained and shown as a solid curve in Fig. 4. The error bar overlaying with the curve represents the $\pm$ one standard deviation on the 100

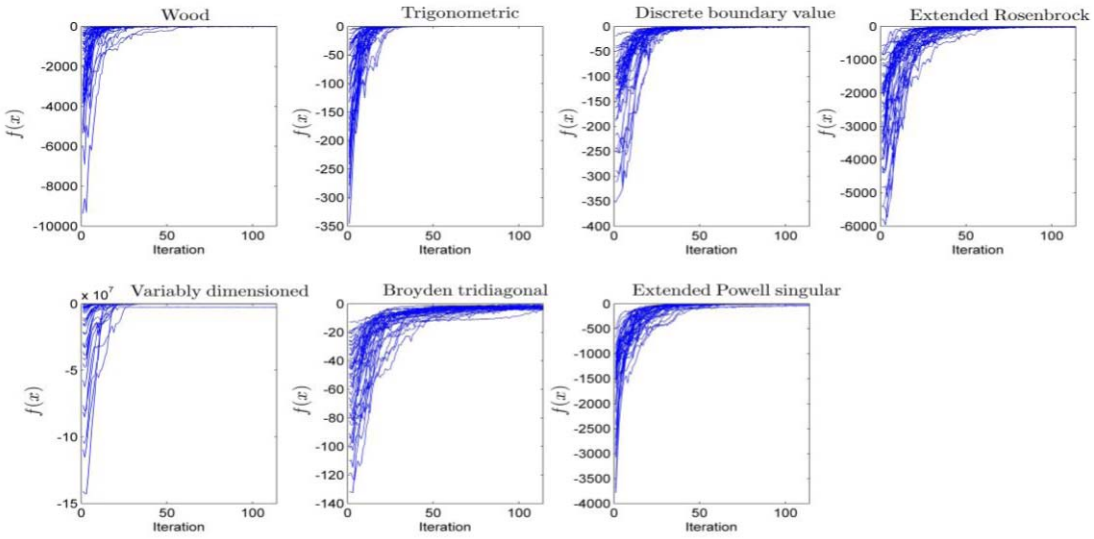

Fig. 3. Improvements in the test function values by BA ($\sigma_\epsilon = 0.01$).

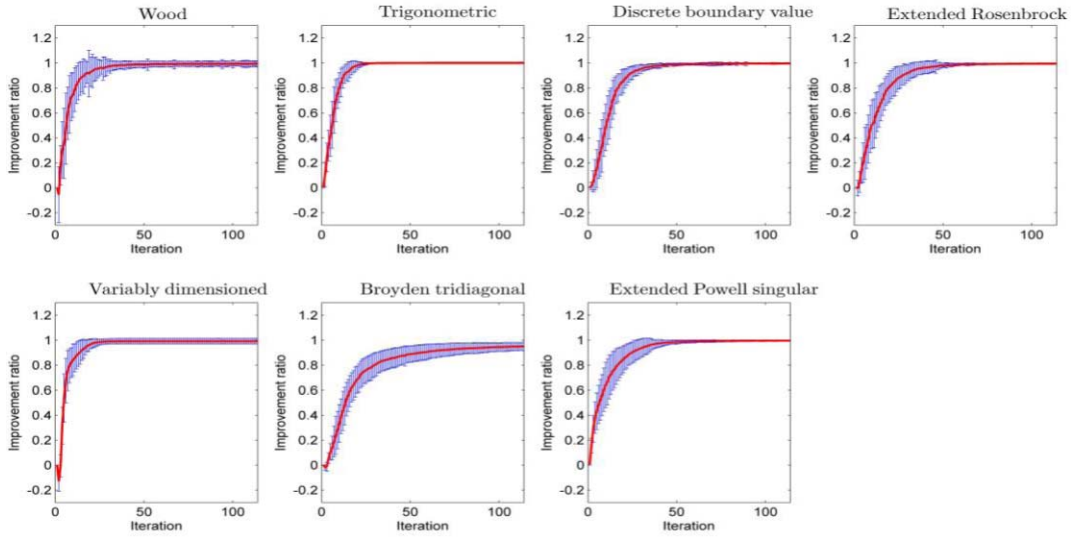

Fig. 4. Normalized improvement in the test function values by BA ($\sigma_\epsilon = 0.01$).

improvement ratios at each iteration. The results show that, except for the Broyden tridiagonal function, the BA algorithm achieves more than 98% of improvement toward the analytical optimum within 50 iterations (50 function evaluations). As for comparison, the derivative-free optimization using surrogate method would require a larger number of function evaluations (ranging from 100 to 300) to achieve similar improvements [25]. Furthermore, the decrease in the standard deviation toward the convergence implies that the optimized outputs by the BA algorithms are almost identical regardless of the initial solutions used. The simulation results show the potential of the BA algorithm to be applied to optimize a complex physical system with a large number of control or optimization variables using the limited amount of input and output data measured from the target system.

Finally, we examine the effect of noise levels on the performance of the BA algorithm. To vary the noise level (randomness) in the noisy output $y = f(\boldsymbol{x})(1 + \epsilon)$, we use three standard deviation values, $\sigma_\epsilon = 0.01, \sigma_\epsilon = 0.03$, and $\sigma_\epsilon = 0.05$, for the Gaussian noise term $\epsilon \sim N(0, \sigma_\epsilon^2)$. For each noise level, 100 simulations of the BA algorithm are conducted per each test function. A total of 700 trajectories of the improvement ratio for the seven test functions are then averaged to construct a single representative performance curve for a given noise level. The three representative improvement ratio curves are compared in Fig. 5. To achieve 90% of improvement in the target function value, 25, 40, and 63 function evaluations are required, respectively, for $\sigma_\epsilon = 0.01, \sigma_\epsilon = 0.03$, and $\sigma_\epsilon = 0.05$. After 100 iterations (function evaluations), the BA algorithm achieves 98.9%, 97.5%, and 93.4% of the improvements, respectively, for $\sigma_\epsilon = 0.01, \sigma_\epsilon = 0.03$, and $\sigma_\epsilon = 0.05$.

As shown in Fig. 5, as the noise level increases, the convergence rate becomes slower and the difference between the converged output by the BA algorithm and the analytical optimum becomes larger. The degradation is caused by the

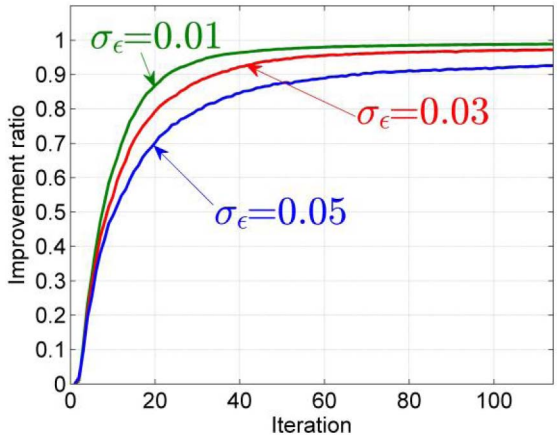

Fig. 5. Normalized improvement in the test function values for different error ratios.

noise added to the sampled function values, which interfere the BA algorithm to identify the ascending direction of the target function. In the BA algorithm, the ascending direction of the target function is probabilistically estimated by observing the function values around the best solution observed so far. When the variations in values of the target function are small compared with the magnitude of the noise added, the estimation becomes less accurate. Estimating the ascending direction can become especially difficult as the algorithm approaches toward the optimum. Near the optimum of a (smooth) target function, the magnitude of the gradient is small, i.e., the local function surface is nearly flat. The nearly flat surface makes it difficult for the BA algorithm to infer the ascending direction, especially when the noise level is large. For moderate error levels, as the results suggest, the BA algorithm can be used to increase the target value as quickly as possible while minimizing the potential cost that can be incurred when sampling data points.

V. Application to Analytical Wind Farm Power Function

The BA algorithm is applied to determine the optimum coordinated control actions of wind turbines in a wind farm with the objective in maximizing the total power production of the wind farm. In a wind farm, wakes formed by the upstream wind turbines decrease the power production of the downstream wind turbines and, therefore, the total wind farm power efficiency. The problem is to adjust the control inputs of the wind turbines to minimize the wake interference among the wind turbines. In this section, the BA algorithm is employed to maximize the analytical wind farm power function, so that the optimum coordinated control actions can be determined by sampling the function values. By comparing the optimized wind farm power efficiencies by the BA algorithm with those by the analytical approach using sequential convex programming as described by [8], we can gain insight into how the BA algorithm performs in solving the cooperative wind farm control problem.

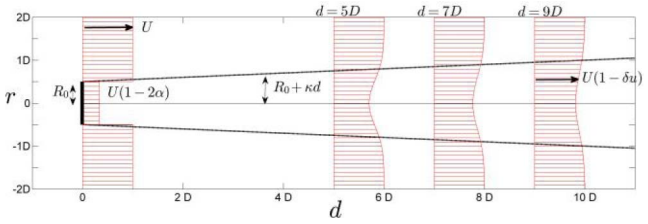

Fig. 6. Continuous wake model [8].

A. Wind Turbine Interaction

Based on the actuator disk model in aerodynamics, the power of a wind turbine due to a wind flow with wind speed U can be expressed as [34]

$$P = \frac{1}{2}\rho A U^3 C_P(\alpha, o) \tag{9}$$

where ρ is the air density and A is the rotor area. $C_P(\alpha, o)$ is termed the power coefficient, which is expressed as [34]

$$C_P(\alpha, o) = \frac{P}{\rho A U^3/2} = 4\alpha(\cos(\beta o) - \alpha)^2 \tag{10}$$

where o denotes the yaw angle offset between the wind direction and the wind turbine rotor, and α is the induction factor representing the relative change between the wind flow speed U and the wind speed right behind the disk.

As the free stream wind flows through a wind turbine, the wind speed $u(d, r, \alpha)$ at the downstream wake distance d and the radial wake distance r in the wake formed behind the wind turbine with an induction factor of α can be expressed as [36]

$$u(d, r, \alpha) = (1 - \delta u(d, r, \alpha))U \tag{11}$$

where $\delta u(d, r, \alpha)$ is the wind speed deficit term quantifying the reduction of the wind speed inside a wake. We treat the deficit factor $\delta u(d, r, \alpha)$ as a continuous function expressed as [8]

$$\delta u(d, r, \alpha) = 2\alpha \left(\frac{R_0}{R_0 + \kappa d}\right)^2 \exp\left(-\left(\frac{r}{R_0 + \kappa d}\right)^2\right) \tag{12}$$

where R_0 is the rotor radius, and κ is the wake expansion coefficient that depends on the surface roughness of a wind farm site. Fig. 6 shows the wind speed profile described by (11) and (12). As shown in Fig. 6, the wind speed recovers (i.e., the deficit factor decreases) as the downstream distance d and the radial wake distance r increase. The power productions of the wind turbines that are near the wake are lowered because of the decrease in wind speed.

B. Cooperative Wind Farm Control Problem

For a given wind speed U and wind direction θ^W on a wind farm with N wind turbines, the energy production function $P_i(\boldsymbol{\alpha}, \boldsymbol{o}; U, \theta^W)$ for wind turbine i can be expressed as a function of the yaw offset angles $\boldsymbol{o} = (o_1, \ldots, o_N)$ and the induction factors $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N)$, which can be adjusted by the blade pitch angle and the generator torque of the wind turbine, as [8]

$$P_i(\boldsymbol{\alpha}, \boldsymbol{o}; U, \theta^W) = \frac{1}{2}\rho A C_P(\alpha_i, o_i)\bar{u}_i^3(\boldsymbol{\alpha}_{-i}, \boldsymbol{o}_{-i}; U, \theta^W) \tag{13}$$

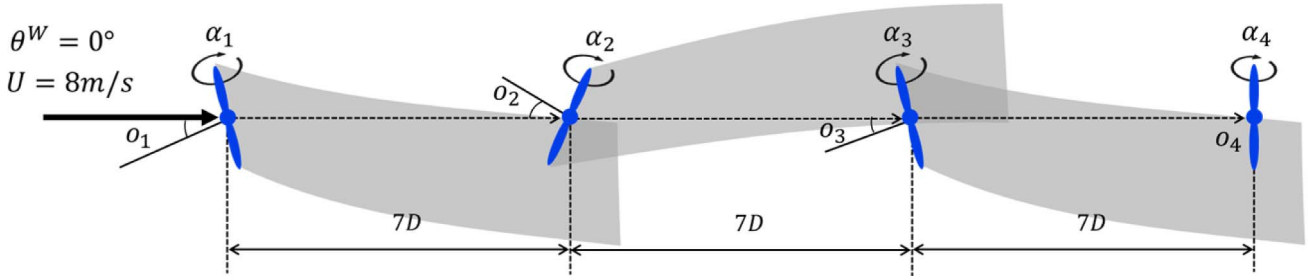

Fig. 7. Numerical simulation of the BA method considering four wind turbines when $\theta^W = 0°$.

where ρ is the air density, A is the rotor area, $C_P(\alpha_i, o_i) = 4\alpha(\cos(o_i) - \alpha_i)^2$ is a power coefficient of wind turbine i representing the power extraction efficiency, $\boldsymbol{\alpha}_{-i} = \boldsymbol{\alpha}\backslash\{\alpha_i\}$ and $\boldsymbol{o}_{-i} = \boldsymbol{o}\backslash\{o_i\}$, and $\bar{u}_i(\boldsymbol{\alpha}_{-i}, \boldsymbol{o}_{-i}; U, \theta^W)$ is the averaged wind speed on the rotor surface of wind turbine i affected by the wake interference of the other (upstream) wind turbines. The yaw offset angle o_i can redirect the wake by changing the direction of thrust force on the rotor, which can reduce the wake overlap on the downstream wind turbine and, thus, increase the power production of the downstream wind turbine. Details of the analytical model, including calibration with the Computational Fluid Dynamics (CFD) simulation data generated using a 5-MW NREL reference wind turbine by SOWFA supercontroller [36], [37], have been described in [8].

For conventional (noncooperative) control strategy, each wind turbine i tries to maximize its own power P_i by manipulating its own control actions, α_i and o_i. Under this control strategy, every wind turbine in the wind farm set $\alpha_i = 1/3$ and $o_i = 0°$ that maximizes (13) regardless of the control actions of other wind turbines [8]. In cooperative control, on the other hand, wind turbines collectively adjust their joint control actions, $\boldsymbol{\alpha}$ and $\boldsymbol{o}$, to maximize the sum of individual wind turbine power, whose control inputs $\boldsymbol{x}$ can be derived by solving the following optimization problem:

$$\max_{\boldsymbol{x}} . f(\boldsymbol{x}; U, \theta^W) \triangleq \sum_{i=1}^{N} P_i(\boldsymbol{\alpha}, \boldsymbol{o}; U, \theta^W) \qquad (14)$$
$$\text{s.t. } \boldsymbol{x}^l \leq \mathbf{x} \leq \boldsymbol{x}^u$$

where $\boldsymbol{x} = (\alpha_i, o_i, \ldots, \alpha_N, o_N)$, and $\boldsymbol{x}^l$ and $\boldsymbol{x}^u$ are, respectively, the lower and upper bounds on the wind turbine control actions for the N wind turbines.

The BA algorithm is applied to find the optimum coordinated control actions $\boldsymbol{x}^* = (\alpha_1^*, o_1^*, \ldots, \alpha_N^*, o_N^*)$ using the least number of function evaluations (sampling power values). The noncooperative control actions, $\alpha_i = 1/3$ and $o_i = 0°$ for $i = 1, \ldots, N$, are used as the initial solution of the BA algorithm. The results of the simulations are represented as the improvement of wind farm power efficiency $\eta(\boldsymbol{x}; \theta^W)$ defined as

$$\eta(\boldsymbol{x}; \theta^W) \triangleq \frac{1}{N} \sum_{i=1}^{N} \eta_i(\boldsymbol{\alpha}, \boldsymbol{o}; \theta^W) \qquad (15)$$

where $\eta_i(\boldsymbol{\alpha}, \boldsymbol{o}; \theta^W) = P_i(\boldsymbol{\alpha}, \boldsymbol{o}; U, \theta^W)/P^*(U)$ is the power efficiency for wind turbine i, with $P^*(U)$ representing the maximum power that can be produced by a wind turbine when there is no wake interference (with the control actions, $\alpha_i = 1/3$ and $o_i = 0°$).

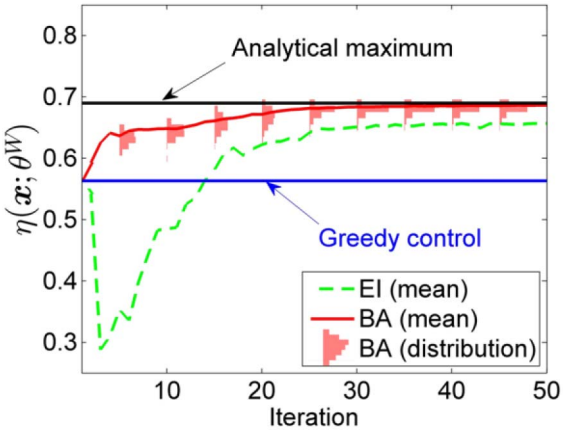

Fig. 8. Trajectories of wind farm power efficiency by BA for different numbers of wind turbines.

C. Simulation Results

The BA algorithm is employed to solve (14) for $N = 4$ and $\theta^W = 0°$. Fig. 7 shows the wind farm configuration for the cooperative wind farm control problem. As shown in Fig. 7, the four wind turbines are arranged in a linear layout separated by $7D$ interdistance (where D is the rotor diameter). Depending on the yaw offset angle and the induction factor, the intensity and the trajectory of a wake varies, and therefore, the power productions of downstream wind turbines are affected. The goal of the cooperative control is to find the optimum induction factors $\boldsymbol{\alpha}^* = (\alpha_1^*, \alpha_2^*, \alpha_3^*, \alpha_4^*)$ and the yaw offset angles $\boldsymbol{o}^* = (o_1^*, o_2^*, o_3^*, o_4^*)$ that maximize the total wind farm power, i.e., the sum of the powers from the four wind turbines. We employ the BA algorithm to find the optimum cooperative control actions using the least number of trial actions (function evaluations). During the simulation, only (o_1, o_2, o_3) and $(\alpha_1, \alpha_2, \alpha_3)$ are optimized while fixing $o_4 = 0°$ and $\alpha_4 = 1/3$, because the noncooperative control action is the unique best action that the last wind turbine can choose to increase the total wind farm power; the deviation from this control action would only decrease its own power production and, thus, the total power production.

Fig. 8 shows the improvement in the total wind farm power efficiency with the iterations of the BA algorithm. A total of 100 optimizations using the BA algorithm are conducted.

For each optimization, the noncooperative control actions, $o_i = 0°$ and $\alpha_i = 1/3$ for $i = 1 \ldots 4$, of the wind turbines are used as the initial control actions for the BA algorithm. The wind farm power efficiency for the noncooperative control strategy is computed using (15) and marked on the plots. In addition, the noisy evaluation of (15), i.e., $y = \eta(\boldsymbol{x}; \theta^W) + \epsilon$ with $\epsilon \sim N(0, \sigma_\epsilon^2)$, is used for the response. Since $\eta(\boldsymbol{x}; \theta^W)$ varies between the noncooperative control efficiency of 0.565 and the analytical maximum efficiency of 0.685, we set $\sigma_\epsilon = 0.003$, which is roughly ~2.5% of the total range of the target function. In spite of using the same initial control actions, the 100 trajectories of the wind farm power efficiency are different due to stochastic nature of the BA algorithm. In Fig. 8, the overall improvement trend is represented using the mean values for the 100-optimization trajectories, which is shown as the solid curve. In addition, the distribution of 100 wind farm power efficiency values at each iteration is shown as histograms overlaying with the mean value curve. Included in Fig. 8 is the analytical maximum of the wind farm power efficiency that is computed by analytically optimizing (14) using sequential convex programming, a mathematical optimization algorithm [8]. For comparison, the mean values for 100 trajectories of wind farm power efficiency obtained by the conventional BO algorithm with the EI acquisition function are also shown in the plot. In this case, the only difference between the BA and BO algorithms in this simulation study is whether the trust region constraint is used or not.

As shown in Fig. 8, BA increases the wind farm power efficiency almost monotonically. The monotonic increase in the wind farm power efficiency is due to the use of the trust region (proximity) constraint that allows sampling the next input only near the best solution observed so far. The effect of trust region constraint is manifested when comparing the efficiency trajectories by the BA algorithm with those by the BO algorithm (with EI acquisition function but without the proximity constraint). Without the trust region constraint imposed in the sampling strategy, the BO algorithm tends to sample the function values over a wide range of input space and often executes trial actions that produce the wind farm power efficiency inferior to that of the noncooperative control (starting point).

The effect of the error level on the efficiency of the BA algorithm is also studied. Fig. 9 compares the trajectories of the wind farm power efficiency obtained by the BA algorithm for different error levels using different values for the standard deviation of the error. As shown in Fig. 9, as the error level increases, the convergence rate becomes slower and the difference between the analytical maximum and the converged efficiency by the BA algorithm becomes larger.

VI. Application to Experimental Wind Farm Control Problem

In Section V, the BA algorithm is employed to optimize the coordinated control actions of wind turbines utilizing an analytically derived wind farm power function. However, constructing such analytical wind farm power function is challenging, in that we need to characterize wind fields, wind turbine blades, a generator, and the interactions among the wind turbines through wakes. As a model-free, data-driven approach, the BA algorithm is designed to improve the total wind farm power efficiency using only the power measurement data from the wind turbines. This section describes an experimental study to validate the capability of the BA algorithm in finding the optimum coordinated control actions using only the power measurement data. For the experiment, the same layout configuration, as shown in Fig. 7, is physically constructed using four scaled wind turbines. The BA algorithm is then applied to maximize the total power productions from the four wind turbines in a wind tunnel laboratory experiment.

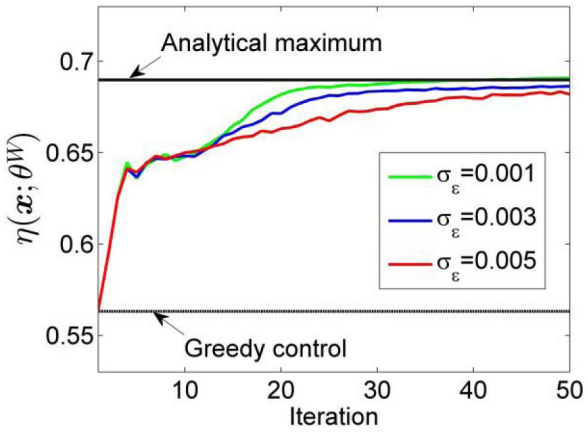

Fig. 9. Trajectories of wind farm power efficiency by BA for different error ratios.

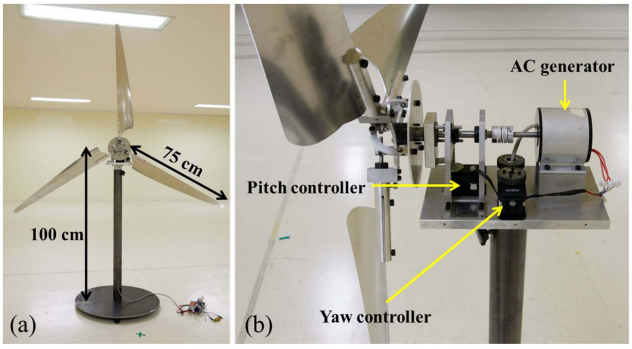

Fig. 10. Scaled wind turbine model. (a) shows the model dimensions and (b) shows the control mechanism.

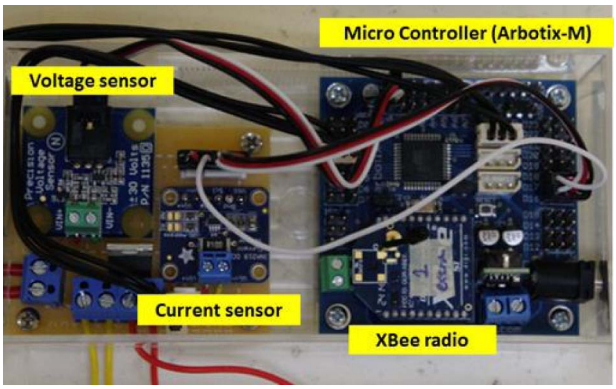

Fig. 11. Control board.

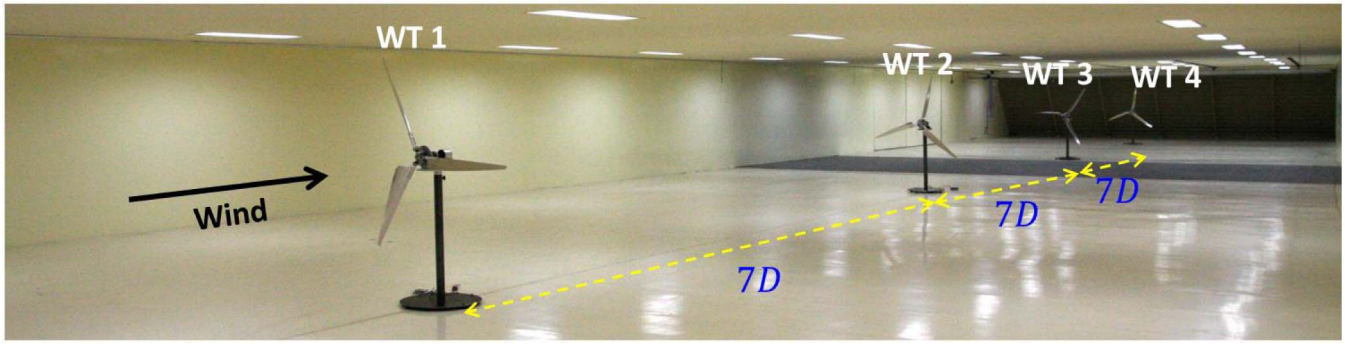

Fig. 12. Layout of the wind turbines in the wind tunnel (KOCED Wind Tunnel Center in Chonbuk National University).

Through this experiment, the feasibility of the BA algorithm for a real-time control application as well as the concept of the cooperative control is tested in a physical setting.

A. Experimental Setup

The scaled wind turbine model, as shown in Fig. 10, is made of three aluminum blades with a length of 70 cm. The rotor diameter is 150 cm. The tower is made of a steel tube with a height of 100 cm. The blade pitch angles are controlled by a servomotor (Dynamixel-64T). As shown in Fig. 10(b), the rotation of the servomotor is transformed into a linear motion to rotate the blade angles through a mechanical linkage. The rotation angles of the servomotor range from 0° to 70° which convert the blade pitch angles varying from 0° to 20° (albeit they are not related in a linear fashion). We use the rotation of the servomotor, instead of the actual blade pitch angle, as the control variable for optimization. The rotational change of the servomotor is easy to track using the encoder in the servomotor, which is also used to acknowledge the executed control actions. As shown in Fig. 10(b), the yaw angle is controlled by the same type of servomotor through a mechanical gear system. With a one-to-one gear ratio, the rotational angle of the servomotor is the same as the actual rotation of the yaw of the wind turbine. An AC generator, as shown in Fig. 10(b), is used to convert the mechanical energy into electrical energy.

Fig. 11 shows the circuit board designed to measure the electrical power output from the wind turbine and to execute the control actions to adjust the blade pitch and yaw angles of the wind turbine. The AC voltage output from the generator is converted into DC voltage by the rectifier. The rectified voltage and the associated current flowing through the load resistance are then measured using voltage and current sensors, from which the instantaneous power is computed. The microcontroller (Arbotix-M) continuously samples the instantaneous power and computes the average power (using a moving average technique). The microcontroller then transmits the computed average power to the central node (laptop computer) through the XBee radio module every 2 min. The BA algorithm processes the average power collected from the wind turbines in the central node and determines the next control actions. The determined control actions are then wirelessly transmitted to the microcontroller to change the blade pitch and the yaw angle in the wind turbine.

Fig. 12 shows the layout of the wind turbines in the wind tunnel experiments. The wind turbines are arranged in a linear pattern and separated by an interdistance of $7D$ (=10.5 m). Using the linear wind farm layout, we study the effectiveness of the BA control algorithm. A constant wind speed of 4 m/s and a wind direction of 0° are used in this experiment. To evaluate the performance of the cooperative control approach and the BA algorithm, two reference wind turbine powers are measured for each wind turbine.

1) P_i^F: Freestream maximum power of wind turbine i that can be produced at a given location when there is no wake interference. The measured power P_i normalized by P_i^F then represents the power efficiency for wind turbine i. The total wind farm power efficiency is computed as $\sum_{i=1}^{N} P_i / \sum_{i=1}^{N} P_i^F$, where N is the number of wind turbines considered.
2) P_i^G: Noncooperative maximum power of wind turbine i that can be produced at a given location when the upstream wind turbines are producing their maximum powers. The wind farm power efficiency for the noncooperative control strategy is then computed as $\sum_{i=1}^{N} P_i^G / \sum_{i=1}^{N} P_i^F$.

For cooperative control, the control actions for the noncooperative optimum are experimentally determined first, from which the BA algorithm proceeds to find the optimum coordinated control actions.

B. Optimization Results

Fig. 13 shows the trajectories of the power efficiency P_i / P_i^F of each individual wind turbine and the associated control actions of the wind turbines with the iterations of the BA algorithm. For cooperative control, the wind turbines collectively adjust their control actions (i.e., the yaw and pitch servo angles) determined by the BA algorithm to increase the total wind farm power production. As shown in Fig. 13, the cooperative control actions lower the power production for the first upstream wind turbine but significantly increase the power productions of the downstream wind turbines. The last downstream wind turbine operates at its noncooperative control actions, since the deviation from the noncooperative control actions only decreases its own power production. It can

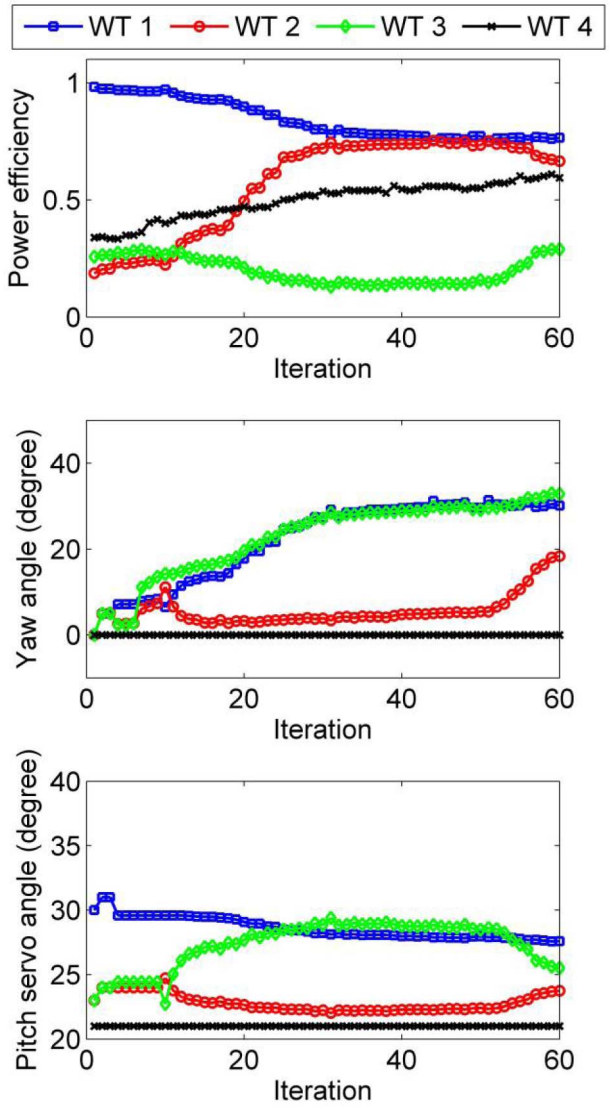

Fig. 13. Control actions and power efficiencies for different numbers of WTs.

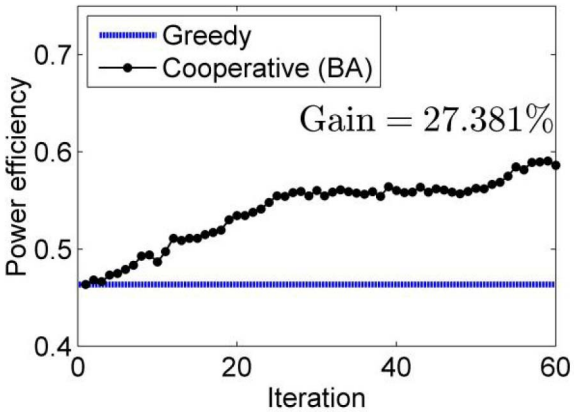

Fig. 14. Improvement on power production using cooperative control for different numbers of WTs.

be seen from Fig. 13 that the gradual modifications on the control actions for the yaw and the pitch are made at each iteration of the BA algorithm. Fig. 14 shows the improvement in the total wind farm power efficiency by the BA algorithm compared with the noncooperative wind farm power efficiency. As shown in Fig. 14, the BA algorithm increases the wind farm power efficiency almost monotonically by gradually changing the control actions of the wind turbines. It should be emphasized that the analytical wind farm power function used in the simulation study is obtained using a specific reference wind turbine calibrated using the data generated from the CFD simulation. Because of the differences in the target wind turbines as well as the wind flow characteristics, the measured power efficiency result from the experiment is quite different from that of the simulated wind farm power efficiency using the analytical wind farm power function. Nonetheless, the trend of wind farm power improvement by the BA algorithm is similar in both the simulation and experiment studies.

VII. Conclusion and Discussion

This paper describes the BA algorithm that can rapidly and almost monotonically find a local optimum of a target system using the limited amount of data. The BA algorithm models the input and output relationship of a target system using GP regression fitted to the measured input and output data. Exploiting the constructed GP model function, the BA algorithm then determines the next sampling point that can best increase the EI. The trust region constraint imposed in the BA algorithm ensures that the next input is selected from the region near the best input observed so far. In addition, the size of the trust region is adaptively adjusted in each iteration to expedite the convergence rate. Due to the proposed sampling strategy, BA is able to increase a target value incrementally with gradual changes in the inputs.

The effectiveness of the BA algorithm in optimizing a target system has been investigated using a set of test functions with input dimensions ranging from 4 to 16. The results show that, for all test functions, the BA algorithm is able to attain 90% improvement in a target value using a small number (i.e., less than 50) of function evaluations. The required number of iterations is usually significantly lower than the number of iterations required for data-driven optimization schemes based on the local surrogate model. Furthermore, the BA algorithm can efficiently optimize the target functions using the noisy function values. The simulation results indicate that, as the noise level increases, the convergence rate decreases and the difference between the converged target function value and the true optimum increases. In general, as shown from the examples, the BA algorithm is robust enough to increase a target value using the measurement data with noise.

To illustrate applicability to real physical problems, the BA algorithm is employed to the cooperative wind farm control problem to maximize the total wind farm power production using only input (control actions) and output (wind farm power production) data. Both simulation and experimental studies have been conducted to test the feasibility of using the BA algorithm in the wind farm control application. For the simulation study, the power output data evaluated using an analytical wind farm power function is used to optimize the control actions. The simulation results show that the

BA algorithm can increase the wind farm power efficiency using a small number of function evaluations. For the experimental study, the power measurement data from the scaled wind turbines are used to determine their optimum control actions that maximize the total power production of the wind turbines. The experimental results show that the BA algorithm is able to increase the total power production by gradually changing the control actions of the wind farm. The simulation and experimental studies suggest that the BA algorithm can be potentially employed to maximize the total power production of a wind farm without constructing an analytical wind farm power function.

For the wind farm power maximization problem, we have assumed that the wind condition remains unchanged during the execution of the BA algorithm. In an actual wind farm site, however, wind condition could continuously change and result in different wake interference patterns. To employ the BA algorithm under varying wind conditions, two approaches are currently being investigated. First is to run multiple BA algorithms in parallel, each of which is designated to maximize the total wind farm power for a certain wind condition; each BA algorithm can be freely interrupted and reinitiated during the variation of wind condition because the GP regression model can memorize the learned target function. Another approach is to treat the wind conditions, i.e., wind speed and wind direction, as random variables and to construct the GP regression model on both the action (wind turbine control inputs) and the context (wind condition). The kernel functions for the wind turbine control actions space and for the wind condition space can be constructed separately and combined to a generalized kernel function for the paired variables between the wind turbine control actions and the wind conditions.

The BA algorithm combines the strengths of BO and trust region optimization to improve a target value rapidly using a limited amount of the input and output data. At the expense of achieving such advantages, the BA algorithm also inherits the weaknesses of these two methods as well. For example, the BA algorithm becomes computationally expensive for fitting the GP regression model as the number of input and output data increases. However, this generally does not cause an issue because the BA algorithm is targeting for optimizing a target system using a small number of input and output points, say less than a hundred. On the other hand, the BA algorithm finds a local optimum that is close to the initial point due to the use of trust region constraint. Noting that there has not yet existed an efficient method that can locate the global optimum using only a small number of data, a good strategy that can find a better (local optimum) solution as quickly as possible, while minimizing the cost associated with the exploration is a desirable alternative for many physical problems. The BA algorithm has great potential as a tool for optimizing the operational conditions of a complex physical system using only the measurement data from the system.

ACKNOWLEDGMENT

The authors would like to thank Prof. S. D. Kwon of Chonbuk National University in Korea, for providing the wind tunnel laboratory (KOCED Wind Tunnel Center) facilities for validating the BA method with the scaled wind turbine models.

REFERENCES

[1] M. S. Adaramola and P.-Å. Krogstad, "Experimental investigation of wake effects on wind turbine performance," *Renew. Energy*, vol. 36, no. 8, pp. 2078–2086, 2011.

[2] D. Medici and J. Å. Dahlberg, "Potential improvement of wind turbine array efficiency by active wake control," in *Proc. Eur. Wind Energy Conf.*, Madrid, Spain, 2003, pp. 65–84.

[3] D. Medici, "Experimental studies of wind turbine wakes—Power optimisation and meandering," Ph.D. dissertation, KTH Roy. Inst. Technol., Stockholm, Sweden, 2005.

[4] N. Marathe, A. Swift, B. Hirth, R. Walker, and J. Schroeder, "Characterizing power performance and wake of a wind turbine under yaw and blade pitch," *Wind Energy*, Jul. 15, 2015, DOI: 10.1002/we.1875

[5] K. E. Johnson and N. Thomas, "Wind farm control: Addressing the aerodynamic interaction among wind turbines," in *Proc. Amer. Control Conf.*, St. Louis, MO, USA, Jun. 2009, pp. 2104–2109.

[6] D. Madjidian and A. Rantzer, "A stationary turbine interaction model for control of wind farms," in *Proc. IFAC 18th World Congr. Int. Fed. Autom. Control*, Milan, Italy, 2011, pp. 4921–4926.

[7] T. Horvat, V. Spudic, and M. Baotic, "Quasi-stationary optimal control for wind farm with closely spaced turbines," in *Proc. 35th Int. Conv. MIPRO*, Opatija, Croatia, May 2012, pp. 829–834.

[8] J. Park and K. H. Law, "Cooperative wind turbine control for maximizing wind farm power using sequential convex programming," *Energy Convers. Manage.*, vol. 101, pp. 295–316, Sep. 2015.

[9] J. R. Marden, S. D. Ruben, and L. Y. Pao, "A model-free approach to wind farm control using game theoretic methods," *IEEE Trans. Control Syst. Technol.*, vol. 21, no. 4, pp. 1207–1214, Jul. 2013.

[10] P. M. O. Gebraad *et al.*, "Wind plant power optimization through yaw control using a parametric model for wake effects—A CFD simulation study," *Wind Energy*, vol. 19, no. 1, pp. 95–114, Jan. 2016.

[11] P. M. O. Gebraad and J. W. Wingerden, "Maximum power-point tracking control for wind farms," *Wind Energy*, vol. 18, no. 3, pp. 429–447, 2015.

[12] C. Kim, Y. Gui, C. C. Chung, and Y.-C. Kang, "A model-free method for wind power plant control with variable wind," in *Proc. IEEE Power Energy Soc. General Meeting*, Washington, DC, USA, Jul. 2014, pp. 1–5.

[13] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvári, "X-armed bandits," *J. Mach. Learn. Res.*, vol. 12, pp. 1655–1695, Jan. 2011.

[14] S. L. Scott, "A modern Bayesian look at the multi-armed bandit," *Appl. Stochastic Models Business Ind.*, vol. 26, no. 6, pp. 639–658, 2010.

[15] H. Tyagi and B. Gärtner, "Continuum armed bandit problem of few variables in high dimensions," in *Proc. 11th Workshop Approx. Online Algorithms (WAOA)*, 2014, pp. 108–119.

[16] V. Kuleshov and D. Precup, "Algorithms for the multi-armed bandit problems," *J. Mach. Learn. Res.*, vol. 1, pp. 1–48, Oct. 2000.

[17] 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.

[18] E. Brochu, V. Cora, and N. Freitas. (2010). "A tutorial on Bayesian optimization of expensive cost functions with application to active user modelling and hierarchical reinforcement learning." [Online]. Available: arXiv:1012.2599

[19] M. Osborne, "Bayesian Gaussian processes for sequential prediction, optimisation and quadrature," Ph.D. dissertation, Dept. Comput. Sci., Univ. Oxford, Oxford, U.K., 2010.

[20] A. Krause and C. S. Ong, "Contextual Gaussian process bandit optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 24. 2011, pp. 2447–2455.

[21] M. J. D. Powell, "UOBYQA: Unconstrained optimization by quadratic approximation," *Math. Program.*, vol. 92, no. 3, pp. 555–582, 2002.

[22] M. J. D. Powell, "Developments of NEWUOA for minimization without derivatives," *IMA J. Numer. Anal.*, vol. 28, no. 4, pp. 649–664, 2008.

[23] K.-H. Chang, L. J. Hong, and H. Wan, "Stochastic trust-region response-surface method (STRONG)—A new response-surface framework for simulation optimization," *INFORMS J. Comput.*, vol. 25, no. 2, pp. 230–243, 2013.

[24] R. Oeuvray and M. Bierlaire, "BOOSTERS: A derivative-free algorithm based on radial basis functions," *Int. J. Model. Simul.*, vol. 29, no. 1, pp. 26–36, 2009.

[25] S. M. Wild, R. G. Regis, and C. A. Shoemaker, "ORBIT: Optimization by radial basis function interpolation in trust-regions," *SIAM J. Sci. Comput.*, vol. 30, no. 6, pp. 3197–3219, 2008.
[26] L. M. Rios and N. V. Sahinidis, "Derivative-free optimization: A review of algorithms and comparison of software implementations," *J. Global Optim.*, vol. 56, no. 3, pp. 1247–1293, 2013.
[27] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.
[28] R. M. Neal, *Bayesian Learning for Neural Networks*. New York, NY, USA: Springer-Verlag, 1996.
[29] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell, "Active learning with Gaussian processes for object categorization," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2007, pp. 1–8.
[30] D. Cox and S. Johns, "SDO: A statistical method for global optimization," in *Multidisciplinary Design Optimization: State of the Art*, N. M. Alexandrov and M.Y. Hussaini, Eds. Philadelphia, PA, USA: SIAM, 1997, pp. 315–329.
[31] J. Mockus, V. Tieshis, and A. Zilinskas, *The Application of Bayesian Methods for Seeking the Extremum* (Toward Global Optimization), L. Dixon and D. Szego, Eds. Amsterdam, The Netherlands: North-Holland, 1978, pp. 117–130.
[32] J. Nocedal and S. Wright, *Numerical Optimization*. New York, NY, USA: Springer, 2000.
[33] J. J. Moré, B. S. Garbow, and K. E. Hillstrom, "Testing unconstrained optimization software," *ACM Trans. Math. Softw.*, vol. 7, no. 1, pp. 17–41, Mar. 1981.
[34] T. Burton, N. Jenkins, D. Sharpe, and E. Bossanyi, *Wind Energy Handbook*, 2nd ed. New York, NY, USA: Wiley, 2011.
[35] N. O. Jensen, "A note on wind generator interaction," Risø DTU Nat. Lab. Sustain. Energy, Roskilde, Denmark, Tech. Rep. Risø-M2411(EN), 1983.
[36] P. Fleming *et al.*, "The SOWFA super-controller: A high-fidelity tool for evaluating wind plant control approaches," in *Proc. EWEA Annu. Meeting*, Vienna, Austria, Feb. 2013.
[37] J. Jonkman, S. Butterfield, W. Musial, and G. Scott, "Definition of a 5-MW reference wind turbine for offshore system development," Nat. Renew. Energy Lab., Golden, CO, USA, Tech. Rep. NREL/TP-500-38060, 2009.

Jinkyoo Park (S'14) received the B.S. degree in architectural engineering from Seoul National University, Seoul, Korea, in 2009, and the M.S. degree in civil and environmental engineering from The University of Texas at Austin, Austin, TX, USA, in 2011. He is currently pursuing the M.S. degree in electrical engineering and the Ph.D. degree in civil and environmental engineering with Stanford University, Stanford, CA, USA.

He has authored or co-authored over 25 articles in journals and conference proceedings. His current research interests include data-driven decision making under uncertainty and its application to engineering systems.

Mr. Park was a recipient of the Best Paper Award of the Manufacturing Engineering Division at the ASME International Manufacturing Science and Engineering Conference in 2015, and the Best Research Paper Award in the area of resilience and smart structures at the ASCE International Workshop on Computing in Civil Engineering in 2013.

Kincho H. Law received the B.S. degree in civil engineering and the B.A. degree in mathematics from the University of Hawaii, Honolulu, HI, USA, in 1976, and the M.S. and Ph.D. degrees in civil engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 1979 and 1981, respectively.

His work has dealt with various aspects of structural sensing, monitoring and control systems, high-performance computing, regulatory and engineering information management, engineering enterprise integration, Internet, and cloud computing. He is currently a Professor of Civil and Environmental Engineering with Stanford University, Stanford, CA, USA. He has authored or co-authored over 400 articles in journals and conference proceedings. His current research interests include engineering informatics and applications of computational science in engineering.

Prof. Law was a recipient of the 2011 American Society of Civil Engineers (ASCE) Computing in Civil Engineering Award. He serves on several editorial boards, including the *ASME Journal of Computing Information and Science in Engineering*, the *ASCE Journal of Computing in Civil Engineering*, and others.