OPERATIONS RESEARCH
Vol. 62, No. 6, November–December 2014, pp. 1416–1438
ISSN 0030-364X (print) | ISSN 1526-5463 (online)

http://dx.doi.org/10.1287/opre.2014.1315

Balancing Exploitation and Exploration in Discrete Optimization via Simulation Through a Gaussian Process-Based Search

Lihua Sun
School of Economics and Management, Tongji University, 200092 Shanghai, China, sunlihua@tongji.edu.cn

L. Jeff Hong
Department of Economics and Finance; and Department of Management Sciences, College of Business, City University of Hong Kong, Kowloon, Hong Kong, jeffhong@cityu.edu.hk

Zhaolin Hu
School of Economics and Management, Tongji University, 200092 Shanghai, China, huzhaolin@gmail.com

Random search algorithms are often used to solve discrete optimization-via-simulation (DOvS) problems. The most critical component of a random search algorithm is the sampling distribution that is used to guide the allocation of the search effort. A good sampling distribution can balance the trade-off between the effort used in searching around the current best solution (which is called exploitation) and the effort used in searching largely unknown regions (which is called exploration). However, most of the random search algorithms for DOvS problems have difficulties in balancing this trade-off in a seamless way. In this paper we propose a new scheme that derives a sampling distribution from a fast fitted Gaussian process based on previously evaluated solutions. We show that the sampling distribution has the desired properties and can automatically balance the exploitation and exploration trade-off. Furthermore, we integrate this sampling distribution into a random research algorithm, called a Gaussian process-based search (GPS) and show that the GPS algorithm has the desired global convergence as the simulation effort goes to infinity. We illustrate the properties of the algorithm through a number of numerical experiments.

Subject classifications: optimization-via-simulation; exploitation and exploration; Gaussian process-based search.
Area of review: Simulation.
History: Received September 2012; revisions received September 2013, May 2014, July 2014; accepted July 2014.
Published online in *Articles in Advance* October 15, 2014.

1. Introduction

Simulation is one of the most widely used operations research tools in practice. When it is used, the goal is often to find the best combination of parameters that optimizes a system performance measure. This is what we call optimization via simulation (OvS). For instance, to design a manufacturing line we need to decide the buffer size of each station to maximize the overall throughput, to manage an inventory system we need to determine the inventory level of all products to maximize the expected profit, and to design an air-cargo terminal we need to choose the number of truck docks for different types of cargos to minimize the average truck waiting time. In the above three examples, the decision variables take integer values. This is called discrete OvS (DOvS). DOvS problems are generally difficult to solve because the objective function is embedded in a stochastic simulation model with no closed-form expressions and can only be estimated through running often time-consuming simulation experiments.

To solve DOvS problems, random search algorithms are often used. Depending on what type of solutions the algorithms converge to, these algorithms can be divided into locally convergent algorithms and globally convergent algorithms. Locally convergent random search algorithms, e.g., the random search of Andradóttir (1995), the COMPASS algorithms of Hong and Nelson (2006) and Hong et al. (2010), the adaptive hyperbox algorithm of Xu et al. (2013), and the R-SPLINE algorithm of Wang et al. (2012), often focus on a local neighborhood to find better solutions. Hong and Nelson (2007) studied this type of algorithm and gave a general framework for local convergence. Although these algorithms often have fast convergence, they can only find local optima. Because the objective functions of DOvS problems are embedded in a black-box simulation model, their convexities are often unknown. Therefore, these algorithms may miss solutions that have significantly better performance than the found local optimum. To partly overcome this drawback, Xu et al. (2010) added a global search phase to identify several good starting points for the COMPASS algorithm to search in the second phase.

There are also many globally convergent random search algorithms, e.g., the stochastic ruler of Yan and Mukai (1992),

the simulated annealing of Alrefaei and Andradóttir (1999), the nested partitions of Shi and Ólafsson (2000), and the SMRAS of Hu et al. (2008). To achieve fast global convergence, these algorithms need to balance the search effort in the current local neighborhood and in the unknown regions. This is known as the exploitation and exploration trade-off. Exploitation refers to the search around the current sample-best solution. Because there are often better solutions near the current sample-best solution, the exploitative search often has a high chance of finding better solutions. Exploration refers to the search in the entire feasible region, especially the regions that are not sufficiently searched. Because there may be regions that are significantly better than the regions that have been searched so far, explorative search may identify those regions and save the search effort that the algorithm may otherwise waste on searching the suboptimal regions. Therefore, how to balance exploitation and exploration is a critical issue in designing globally convergent random search algorithms.

Globally convergent random search algorithms in the literature can be divided into four classes—exploitation-based, exploration-based, combined, and integrated—based on their approaches to handling the exploitation and exploration trade-off. Exploitation-based algorithms, e.g., the stochastic ruler of Yan and Mukai (1992), the stochastic comparison of Gong et al. (1999), and the simulated annealing of Alrefaei and Andradóttir (1999), only search the local neighborhood of the current sample-best solution, but they may move to an inferior solution with certain probabilities to achieve global convergence. Exploration-based algorithms, e.g., the random search algorithm of Andradóttir (1996), search in the entire region randomly to avoid being trapped in a suboptimal region, but they may miss easy opportunity of finding a better solution in the neighborhood of the current sample-best solution.

Many random search algorithms in the literature take into consideration the trade-off between exploitation and exploration and determine a sampling distribution that allows all solutions to be sampled but allocate higher probabilities to solutions in promising regions. In these kinds of algorithms, the sampling points can "jump" to a better region rather than "move" to a better region in an exploitation-based algorithm. We divide these algorithms into combined and integrated algorithms based on how they handle the trade-off. Combined algorithms use a predetermined scheme to balance the trade-off, while integrated algorithms balance the trade-off adaptively based on the available information in the optimization process. Combined algorithms typically focus on exploitative search while either adding a fixed amount of effort in each iteration or assigning a fixed sequence of iterations to conduct explorative search. For instance, the nested partitions algorithm of Pichitlamken and Nelson (2003) uses the first method, and the BEESE framework of Andradóttir and Prudius (2009) considers both. The integrated algorithms typically have an integrated sampling distribution governing the search effort in each iteration instead of separating the exploitation and exploration as in the combined algorithms. They update the sampling distribution in each iteration based on the available information. For instance, the SMRAS algorithm of Hu et al. (2008) uses a parameterized sampling distribution and updates the parameters of the distribution by minimizing the Kullback-Leibler divergence based on the elite solutions in each iteration. When the distribution is chosen properly, the algorithm can balance exploitation and exploration and converge to the global optimum.

In this paper, we propose another integrated random search algorithm called the Gaussian process-based search (GPS) algorithm. In each iteration of the algorithm, we construct a Gaussian process whose mean function passes through the sample means of all previously evaluated solutions. Based on the constructed Gaussian process, the probability that a solution is better than the current sample-best solution may be calculated. The GPS algorithm then builds a sampling distribution based on these probabilities (without calculating them explicitly), and samples some new solutions from this distribution. Similar to the SMRAS, the GPS algorithm automatically balances the trade-off between exploitation and exploration. Different from the SMRAS, it does not use a parameterized sampling distribution family, and it updates the sampling distribution based on the performance and locations of all previously simulated solutions instead of only the elite solutions as in SMRAS.

Surface fitting using Gaussian process is also known as kriging. It was first proposed in the field of geostatistics and has been studied extensively since then. The readers may refer to Biles et al. (2007) and Ankenman et al. (2010) for more details on kriging metamodeling for stochastic simulation experiments. In typical kriging techniques, the goal is to fit the surface given a set of observations on the surface, either with or without estimation errors. Therefore, these techniques often try to minimize the integrated mean square error (IMSE) of the fitted surface. To achieve the minimal IMSE, these techniques need to invert a $n \times n$ covariance matrix where n is the number of simulated solutions, or take a Cholesky decomposition of this matrix and solve a system of linear equations. These operations may require a significant amount of computation if the number of solutions is large and may encounter numerical problems if there are solutions that are very close to each other (which makes the matrix ill-conditioned). In kriging examples, this matrix inversion rarely causes serious problems because the number of solutions is often small and these solutions are typically well spread. In the GPS algorithm, however, the number of solutions that the algorithm visits grows fast as the number of iterations increases, and the solutions tend to cluster around some good ones. Hence, the covariance matrix is often of large size and ill-conditioned. Furthermore, a Gaussian-process fitting needs to be done in each iteration of the random search algorithm and,

thus, cannot be computationally expensive. Therefore, *kriging techniques requiring matrix inversions are generally computationally not applicable to our algorithm*. Chapter 8 of Rasmussen and Williams (2006) provides a thorough discussion on the various approximation techniques to conduct matrix inversion. However, these techniques are themselves quite complicated and computationally intensive. Therefore, they are also not suitable for kriging-based search,which requires surface fitting in each iteration and requires hundreds, even thousands, of iterations to find the global optimum.

In this paper we carefully analyze the necessary properties that allow a Gaussian process to yield a good sampling distribution for search algorithms. We find that the minimal IMSE is not one of them, because our goal is not to fit an accurate surface but to construct an appropriate sampling distribution. Then, we design a new technique to construct a Gaussian process that maintains all the necessary properties but requires no matrix inversion for surface fitting. It is computationally very efficient even when the number of evaluated solutions is very large and the evaluated solutions are clustered, yet it yields a sampling distribution that nicely balances the exploitation and exploration trade-off.

Gaussian processes have also been used in other types of black-box optimizations. An important class is the optimization problems where the function evaluations are very expensive (often more expensive than the type of problems we are interested in). In this area, the two most famous algorithms are the *P*-algorithm (Kushner 1964 and Calvin and Žilinskas 1999), which is originally designed for one-dimensional problems, and the efficient global optimization (EGO) algorithm (Jones et al. 1998), which could be applied for solving multidimensional problems. Both algorithms use standard kriging techniques to fit the surface, and they solve an often-difficult global optimization problem to find the next solution for simulation by maximizing either the probability of being delta-better than the current best solution or the expected improvement from the current best solution. These algorithms work well for black-box optimization problems where only a small number of function evaluations can be done. Similar ideas and algorithms have also been suggested to handle general multidimensional stochastic optimization problems as well; see, for instance, Sasena (2002) (a nice review of the *P*-algorithm and how researchers have extended it to multidimensional settings can be found in §§2.3.1 and 2.3.2 therein), Huang et al. (2006), and Quan et al. (2013). Nevertheless, these algorithms also have some limitations. First, they cannot efficiently handle (in fact, are not designed to handle) the type of problems that we are interested in, where the simulation outputs are stochastic and we may afford thousands to millions of simulation observations. Second, it is often difficult to find the next solution to simulate especially when the number of visited solutions is large, because a multidimensional nonconvex global optimization problem needs to be solved in each iteration. One can refer to Jones et al. (1998) for details about the difficulties. Our algorithm shares the similarity with the *P*-algorithm in that we also consider the probability of being better than the current best solution. But starting from the probability quantities, our approach becomes different, and we think it is much more suitable for typical DOvS problems. In §5 of this paper, we compare our algorithm with the sequential kriging optimization (SKO) algorithm proposed by Huang et al. (2006) and demonstrate the advantages of our algorithm in solving DOvS problems.

The rest of this paper is organized as follows. In §2, we discuss the desired properties of a sampling distribution and illustrate our idea using a one-dimensional deterministic example. In §3, we propose a fast scheme to fit a Gaussian process for multidimensional cases and discuss how to sample from the derived sampling distribution. In §4, we give a detailed description of the GPS algorithm, prove its global convergence, and propose stopping criteria that may be used in the algorithm. Numerical results are reported in §5, followed by the conclusions in §6.

2. Desired Properties of Sampling Distribution

We are interested in solving DOvS problems with the following form:

$$\max_{\mathbf{x}\in\Theta} g(\mathbf{x}) := \mathrm{E}[G(\mathbf{x})] \tag{1}$$

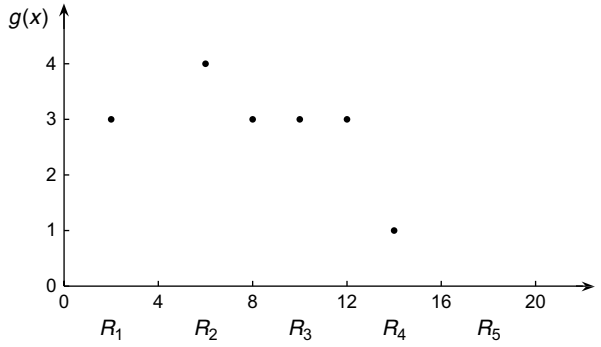
where the random variable $G(\mathbf{x})$ typically has no closed-form expression, but can be observed through running a simulation experiment at $\mathbf{x}$, and the solution set Θ is a finite discrete set. In particular, we assume that $\Theta = \Omega \cap \mathcal{I}^d$, where $\Omega \subset \Re^d$ is a convex compact set and $\mathcal{I}^d$ is the set of d-dimensional integer vectors. Therefore, Θ contains all integer solutions in a convex compact set Ω. Throughout the paper we assume that $\mathrm{E}[G^2(\mathbf{x})] < \infty$ for all $\mathbf{x} \in \Theta$, which also implies that $\mathrm{E}[G(\mathbf{x})] < \infty$ for all $\mathbf{x} \in \Theta$. Let g^* denote the maximal value of g on Θ. Because Θ is a finite discrete set, g^* is finite and there exists at least one point $\mathbf{x}^* \in \Theta$ such that $g(\mathbf{x}^*) = g^*$.

Random search algorithms are often used to solve Problem (1). In each iteration of a random search algorithm, a sampling distribution needs to be constructed, and new solutions are sampled based on this distribution. Selection of a proper sampling distribution is central to the trade-off between exploitation and exploration and is critical to the performance of the algorithm. In this section, we discuss the desired properties of sampling distributions and illustrate our idea of constructing a proper sampling distribution using a simple one-dimensional deterministic example.

2.1. Desired Properties of a Sampling Distribution

Suppose that we want to solve a one-dimensional global maximization problem whose objective function $g(x)$ has no closed-form expression but can be evaluated without

Figure 1. Function values of g at some points.



noise through a deterministic simulation experiment at x for any $x \in [0, 20]$. Through the current iteration of a random search algorithm, we have evaluated six solutions $x_1, \ldots, x_6$, as shown in Figure 1. Based on the given information, how should the sampling probabilities be allocated to all feasible solutions?

For clear illustration, we divide the feasible region into five subregions, denoted as $R_1, \ldots, R_5$. We believe that a reasonable sampling distribution for this example should have at least the following properties:

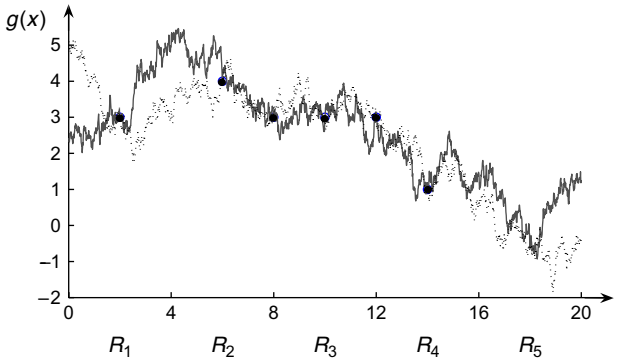
- allocating higher probabilities to solutions in R_2 because it is likely to find better solutions around the current best one;
- allocating higher probabilities to R_1 than to R_3, even though the evaluated solutions in these two regions have the same objective value, because R_1 is less explored;
- allocating higher probabilities to R_5 than R_4 because R_4 contains the current worst solution and R_5 has not been explored yet.

Although all these properties make sense, to the best of our knowledge none of the existing random search algorithms can construct a sampling distribution that satisfies all these properties. In the remainder of this section we show how to construct such a sampling distribution.

2.2. The Idea of Building a Sampling Distribution

Suppose that $g(x)$ is a sample path of a Brownian motion process $B(x + a)$, where $a > 0$ is a very large number, e.g., we may set $a = 10^9$ in this example. We let $Y(x) = B(x + a)$. Notice that, when there is no observations of $g(x)$, $Y(x)$ has a very large variance around its mean $\mathrm{E}[Y(x)] = 0$ for any $x \in [0, 20]$. The large variance is used to illustrate that we have no information on where the function $g(x)$ is located. After we have collected six observations, i.e., $(x_1, g(x_1)), \ldots, (x_6, g(x_6))$, we know that the Brownian motion process passes through these points. This restricts what the sample path may possibly look like in the feasible region (Figure 2). Conditioning on this restriction, we may derive the conditional means and variances of all feasible solutions based on the properties of the Brownian motion process (see Figure 3). Then, we can further calculate the conditional probability that each solution has

Figure 2. Two sample paths of a Brownian motion process passing through $x_1, \ldots, x_6$.



a value that is greater than the value of the current best solution, i.e., $\Pr^*\{Y(x) > 4\}$ in this example (see Figure 4), where the "*" notation denotes it is a conditional statement conditioned on all available information. We propose to normalize these conditional probabilities, i.e.,

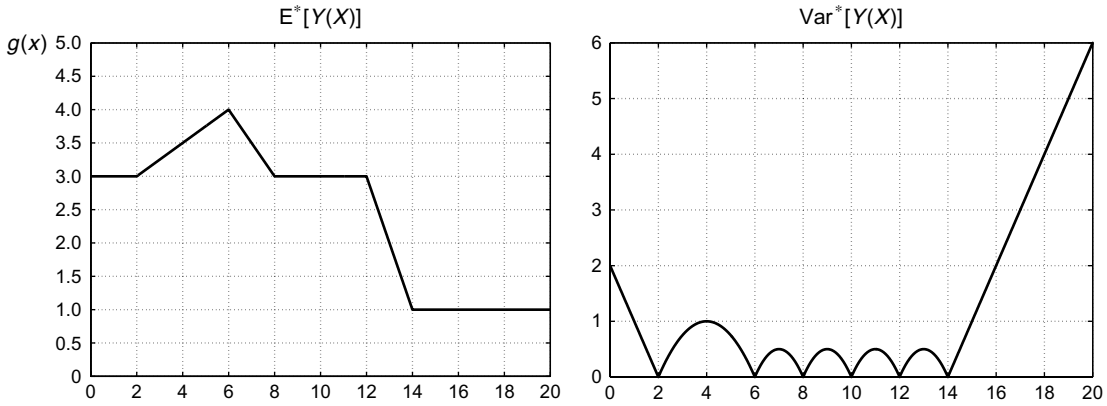
$$f(x) = \frac{\Pr^*\{Y(x) > 4\}}{\sum_{z \in \Theta} \Pr^*\{Y(z) > 4\}}, \tag{2}$$

and use $f(\cdot)$ as a sampling distribution. From Figure 4, it is clear that this sampling distribution satisfies all the desired properties listed in §2.1.

To explain the reason why the proposed approach can generate a sampling distribution that satisfies the desired properties, we note that the approach takes a somewhat informal Bayesian viewpoint. Our prior belief (as well as the prior beliefs of many others) is that the objective function $g(x)$ displays a certain level of continuity, suggesting that the solutions around good solutions tend to be good, while the solutions around bad solutions tend to be bad. In our approach the Brownian motion process, as a continuous random process, is used to capture such continuity. The Brownian motion process also automatically updates its shape to match the true surface by incorporating the information provided by the evaluated solutions (i.e., $x_1, \ldots, x_6$ and their function values in our example). Such a Bayesian approach has a deep root in the learning literature. In the learning context, a prior model, e.g., a Gaussian process, is first assumed. After collecting some observations of the function values, a posterior model is derived using a Bayesian approach to incorporate the new information, and predictions are then made based on the posterior model. For a thorough discussion on the Bayesian approach in the learning context, readers are referred to Rasmussen and Williams (2006).

Notice that, under our Brownian motion model, solutions around the current best solution typically have high conditional *mean* values, thus, may offer high conditional probabilities of being better than the current best solutions; and solutions in largely unexplored regions typically have high conditional *variances*, thus, may offer high conditional probabilities as well. These probabilities naturally

Figure 3. Conditional mean and variance of the Brownian motion process.



balance the trade-off between exploitation and exploration, and their normalized distribution is a good candidate of a sampling distribution. Therefore, in our approach, the conditional mean and variance of a new solution reflect the solution's potential for exploitative search and explorative search, respectively, and the conditional probability that the solution is better than the current best solution balances naturally the exploitation and exploration trade-off.[1]

It is also worthwhile to note that the nondifferentiability of the Brownian motion process does not impose any restriction on our approach. In our approach, the Brownian motion process is only used to generate a sampling distribution that can balance the exploitation and exploration trade-off, and it does not have to match the smoothness of $g(x)$. Indeed, one may also consider other one-dimensional Gaussian processes for this example. For instance, the Ornstein-Uhlenbeck process may also be applied to derive the conditional means and variances for all the feasible solutions, and the resulting sampling distribution often has a similar shape as the one constructed using the Brownian motion process.

However, the (one-dimensional) Brownian motion process that we use in this section can only be applied to one-dimensional deterministic optimization problems. To apply this approach to multidimensional DOvS problems, we need to consider the following issues:

- how to extend the idea from one-dimensional problems to multidimensional ones?
- how to handle the noise in the estimates of objective values?
- how to sample efficiently from the sampling distribution without calculating the normalizing constant, i.e., the denominator in Equation (2)?

We discuss all these issues in §3 and discuss how to design DOvS algorithms based on the sampling distribution in §4.

3. Gaussian Process-Based Sampling Distribution

In the previous section we use a one-dimensional deterministic example to illustrate our idea of constructing a sampling distribution that balances the exploitation and exploration trade-off. In this section we extend the idea to higher dimensional surfaces with estimation errors. We first discuss how to fit response surfaces and construct sampling distributions based on traditional kriging metamodeling techniques and discuss their limitations in solving DOvS problems. We then propose a novel approach to surface fitting and discuss its advantages.

3.1. Kriging-Based Search

The traditional kriging method models the simulation output at a point $\mathbf{x}$ as

$$G(\mathbf{x}) = M(\mathbf{x}) + \epsilon(\mathbf{x}), \tag{3}$$

where $M(\mathbf{x})$ is a stationary Gaussian process with mean 0 and covariance function $\sigma^2\gamma(\cdot,\cdot)$, and $\epsilon(\mathbf{x})$ is a normal random variable with mean 0 and variance $\sigma_\epsilon^2(\mathbf{x})$.

In Equation (3), the stationary Gaussian process $M(\mathbf{x})$ models the unknown objective function $g(\mathbf{x})$, and its correlation function $\gamma(\mathbf{x}_1,\mathbf{x}_2) = \mathrm{Corr}(M(\mathbf{x}_1), M(\mathbf{x}_2))$ is typically a function of $\|\mathbf{x}_1 - \mathbf{x}_2\|$, where $\|\cdot\|$ denotes the Euclidean distance. Let

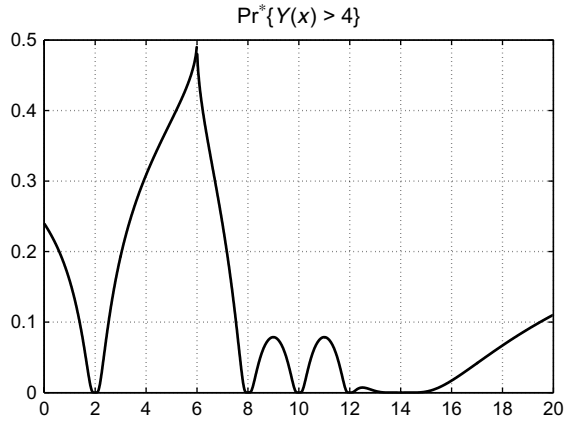
$$\gamma(\mathbf{x}_1,\mathbf{x}_2) = h(\|\mathbf{x}_1 - \mathbf{x}_2\|). \tag{4}$$

Depending on the functional forms of $h(\cdot)$, the sample path of $M(\mathbf{x})$ may display different levels of differentiability (Santner et al. 2003). For instance, when $h(t) = \exp(-at^2)$ for some positive constant a, the sample path generated by $M(\mathbf{x})$ is infinitely differentiable.[2] Throughout this paper we require the correlation function $h(\cdot)$ to satisfy the following condition.

Condition 1. The correlation function $0 \leqslant h(t) \leqslant 1$ is a decreasing function of t when $t \geqslant 0$ and, for any $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2 \in \Theta$, $h(\|\mathbf{x}_1 - \mathbf{x}_2\|) \geqslant h(\|\mathbf{x}_0 - \mathbf{x}_1\|) \cdot h(\|\mathbf{x}_0 - \mathbf{x}_2\|)$.

Many correlation functions satisfy Condition 1. For instance, the exponential correlation function $h(t) =$

Figure 4. $\Pr^*\{Y(x) > 4\}$ under the Brownian motion process.



$\exp(-at)$ with $a > 0$ and the Gaussian correlation function $h(t) = \exp(-at^2)$ with $a > 0$ both satisfy the condition. In these two examples, Condition 1 becomes a triangle inequality in the solution space.

In Equation (3), the error term $\epsilon(\mathbf{x})$ models the noise in the simulation output, and it can handle the heteroscedasticity of simulation outputs at different $\mathbf{x}$ values. Notice that $M(\mathbf{x})$ and $\epsilon(\mathbf{x})$ capture completely different sources of randomness; it is reasonable to assume that they are independent of each other. Furthermore, as we only consider independent simulation experiments at different $\mathbf{x}$ values in this paper, we have $\mathrm{Cov}(\epsilon(\mathbf{x}), \epsilon(\mathbf{x}')) = 0$ for any $\mathbf{x} \neq \mathbf{x}'$.

Suppose that, through the current iteration, a random search algorithm has visited m points, denoted as $\mathbf{x}_1, \ldots, \mathbf{x}_m$, and have taken n_i simulation replications for $\mathbf{x}_i$, $i = 1, \ldots, m$. Let $\bar{G}(\mathbf{x}_i)$ denote the sample mean calculated from n_i observations of $G(\mathbf{x}_i)$ for all $i = 1, \ldots, m$, let $\bar{\mathbb{G}} = (\bar{G}(\mathbf{x}_1), \ldots, \bar{G}(\mathbf{x}_m))^{\mathrm{T}}$, and let Σ_ϵ be an $m \times m$ matrix whose (i,i)th element is $\sigma_\epsilon^2(\mathbf{x}_i)/n_i$ whereas all other elements are 0. Notice that Σ_ϵ is the covariance matrix implied by the intrinsic noise (Ankenman et al. 2010). Furthermore, we let Γ be an $m \times m$ matrix whose (i,j)th element is $\gamma(\mathbf{x}_i, \mathbf{x}_j)$, and $\gamma(\mathbf{x}_0)$ be an m-dimensional vector whose ith element is $\gamma(\mathbf{x}_0, \mathbf{x}_i)$. Then, Ankenman et al. (2010) show that for any $\mathbf{x}_0$, the MSE-optimal linear predictor of $g(\mathbf{x}_0)$ conditioned on the observed data is

$$\hat{g}(\mathbf{x}_0) = \lambda(\mathbf{x}_0)^{\mathrm{T}} \bar{\mathbb{G}} \tag{5}$$

with

$$\lambda(\mathbf{x}_0)^{\mathrm{T}} = \gamma(\mathbf{x}_0)^{\mathrm{T}} \left(\Gamma + \frac{1}{\sigma^2} \Sigma_\epsilon \right)^{-1}, \tag{6}$$

and the corresponding optimal MSE is

$$\mathrm{MSE}(\mathbf{x}_0) = \sigma^2 [1 - \lambda(\mathbf{x}_0)^{\mathrm{T}} \gamma(\mathbf{x}_0)]. \tag{7}$$

Remark 1. The MSE-optimal predictor is also called the stochastic kriging model. If there is no noise in the simulation outputs, we may set $\sigma_\epsilon^2(\mathbf{x}) \equiv 0$. Then, the model degenerates to a deterministic kriging model (Stein 1999).

The results generated from stochastic kriging via MSE-optimal prediction may also be derived or interpreted from a Bayesian perspective. In particular, Rasmussen and Williams (2006) assume that $\Sigma_\epsilon = \sigma_m^2 I$ where I is an identity matrix, and they show that if the prior is a Gaussian process, then the posterior distribution given the acquired information turns out to be a normal distribution with mean and variance exactly given by (5) and (7), respectively (with, of course, $\Sigma_\epsilon = \sigma_m^2 I$); readers may refer to Chapter 2 of Rasmussen and Williams (2006) for the detailed derivations and discussions.

For every $\mathbf{x} \in \Theta$, let $Y'(\mathbf{x})$ denote the metamodel of $g(\mathbf{x})$ in the stochastic kriging model. The above analysis shows that, conditioned on all available information (i.e., all the observations of all the visited solutions), $Y'(\mathbf{x})(= M(\mathbf{x}))$ is normally distributed with $\mathrm{E}^*[Y'(\mathbf{x})] = \hat{g}(\mathbf{x})$ and $\mathrm{Var}^*[Y'(\mathbf{x})] = \mathrm{MSE}(\mathbf{x})$, where we use the notation $\mathrm{E}^*(\cdot)$, $\mathrm{Var}^*(\cdot)$ and $\Pr^*(\cdot)$ to denote they are conditioned on all available information. Similar to the one-dimensional deterministic example reported in §2, the mean function $\hat{g}(\mathbf{x})$ measures the need for exploitation and the variance function $\mathrm{MSE}(\mathbf{x})$ measures the need for exploration. Let c denote the current sample-best value, i.e., $c = \max\{\bar{G}(x_1), \ldots, \bar{G}(x_m)\}$. Then, the conditional probability $\Pr^*\{Y'(\mathbf{x}) > c\}$ may be calculated, and we may define the sampling distribution as

$$f(\mathbf{x}) = \frac{\Pr^*\{Y'(\mathbf{x}) > c\}}{\sum_{\mathbf{z} \in \Theta} \Pr^*\{Y'(\mathbf{z}) > c\}}, \quad \mathbf{x} \in \Theta. \tag{8}$$

Notice that, for any $\mathbf{x} \in \Theta$, $\Pr^*\{Y'(\mathbf{x}) > c\}$ represents the conditional probability that $\mathbf{x}$ has a value that is better than the current sample-best value c and $f(\mathbf{x})$ represents the relative importance of $\mathbf{x}$ among all solutions in Θ in its probability of being a better solution. Based on the discussions in §2, $f(\mathbf{x})$ automatically balances the exploitation and exploration trade-off.

3.2. Fast Construction of Gaussian Process

Although the sampling distribution defined by Equation (8) is appealing, its implementation in random search algorithms may be problematic because of the matrix inversion operation needed in Equation (6). Notice that kriging techniques are originally designed for surface fitting instead of optimization. When fitting a surface, the parameters of the Gaussian process are often estimated using the maximum likelihood estimation method, and the prediction of the value at a given solution is determined through minimizing its MSE. In such cases, the number of observed solutions is typically not large, and the points are typically well spread in the solution space. Then, the matrix $\Gamma + (\sigma^2)^{-1}\Sigma_\epsilon$ is poorly conditioned, and inverting it is often not a difficult numerical problem; even if it is, the inversion is only conducted once. When Gaussian processes are used in random search algorithms, however, we are in a different situation. In random search algorithms, Gaussian processes need

to be constructed repeatedly, the numbers of solutions are often quite large, and some of the solutions are close to each other (because random search algorithms often simulate solutions that are clustered around some good ones). Our numerical experiments show that the matrix inversion may become prohibitive when the matrix becomes large. This leads to serious implementation issues when kriging techniques of Equations (5) to (8) are used to construct sampling distributions.[3]

The problem of matrix inversion in Gaussian process prediction has been studied in the machine learning literature. When the dimension of the matrix is large and the matrix is poorly conditioned, methods have been proposed to avoid a direct matrix inversion. For instance, one may compute the weights defined in Equation (6) by conducting a Cholesky decomposition and then solving a system of linear equations (see, e.g., Algorithm 2.1 in Rasmussen and Williams 2006). However, the Cholesky decomposition approach does not resolve the difficulty of singular or near-singular matrices. To tackle the issues of large dimension and possible singularity caused by a large data set, many techniques have been proposed for Gaussian process prediction (see, Chapter 8 of Rasmussen and Williams 2006 for a thorough discussion on the various techniques). Nevertheless, these techniques only provide approximate solutions by, for instance, approximating the eigenvalues or using only a part of the data points. Furthermore, these techniques are themselves quite complicated and computationally intensive. For random search algorithms, the accuracy of the approximation is often not a major concern, but the computational efficiency is. Therefore, these approximations are not suitable to be used repeatedly in an optimization algorithm. Because of these issues, we choose to develop a new method of fitting Gaussian processes that is fast yet maintains the desired properties of the sampling distributions.

To develop a new method of fitting Gaussian processes used in random search algorithms, we only need to develop a model of $Y'(\mathbf{x})$ such that $\mathrm{E}^*[Y'(\mathbf{x})]$ and $\mathrm{Var}^*[Y'(\mathbf{x})]$ represent the necessities of exploitation and exploration, while the quality of surface fitting is not critical. This is because of two reasons. First, good $\mathrm{E}^*[Y'(\mathbf{x})]$ and $\mathrm{Var}^*[Y'(\mathbf{x})]$ functions are all that we need to derive good sampling distributions that balance the exploitation and exploration tradeoff. Second, solutions sampled from the derived sampling distributions still contain a significant amount of randomness (due to random sampling); it becomes unnecessary to minimize the MSE of the prediction. Notice that the techniques of kriging and stochastic kriging have deep roots in Bayesian analysis and often follow a rigorous Bayesian framework. In our case, however, we take a more pragmatic viewpoint and use the Gaussian process models only to derive good $\mathrm{E}^*[Y'(\mathbf{x})]$ and $\mathrm{Var}^*[Y'(\mathbf{x})]$ functions.

Suppose that $M(\mathbf{x})$ and $\bar{\mathbb{G}}$ are defined as in §3.1. In contrast to the MSE-optimal metamodel $Y'(\mathbf{x})$, we propose the following model to model $g(\mathbf{x})$:

$$Y(\mathbf{x}) = M(\mathbf{x}) + \lambda(\mathbf{x})^{\mathrm{T}}(\bar{\mathbb{G}} - \mathbb{M}) + \lambda(\mathbf{x})^{\mathrm{T}}\mathscr{E}, \tag{9}$$

where $\lambda(\mathbf{x}) = (\lambda_1(\mathbf{x}), \ldots, \lambda_m(\mathbf{x}))^{\mathrm{T}}$ is a vector of weight functions, $\mathbb{M} = (M(\mathbf{x}_1), \ldots, M(\mathbf{x}_m))^{\mathrm{T}}$ is a vector of $M(\mathbf{x})$ evaluated at $\mathbf{x}_1, \ldots, \mathbf{x}_m$, and $\mathscr{E} = (\epsilon_1, \ldots, \epsilon_m)^{\mathrm{T}}$ is an m-dimensional random vector following a multivariate normal distribution with mean $\mathbf{0}$ and covariance matrix

$$\Sigma_{\mathscr{E}} = \mathrm{diag}\left\{\frac{\sigma^2(\mathbf{x}_1)}{n_1}, \ldots, \frac{\sigma^2(\mathbf{x}_m)}{n_m}\right\}. \tag{10}$$

Furthermore, $M(\mathbf{x})$, $\bar{\mathbb{G}}$ and $\mathscr{E}$ are mutually independent of each other. The model in Equation (9) uses three parts to model the different aspects of $g(\mathbf{x})$. It uses a stationary Gaussian process $M(\mathbf{x})$ to capture the continuity and the uncertainty of $g(\mathbf{x})$ when there is no additional information, it incorporates the information from past observations at $\mathbf{x}_1, \ldots, \mathbf{x}_m$ in the term $\lambda(\mathbf{x})^{\mathrm{T}}(\bar{\mathbb{G}} - \mathbb{M})$, and it captures the randomness in $\bar{G}(x_i)$, $i = 1, \ldots, m$, by term $\lambda(\mathbf{x})^{\mathrm{T}}\mathscr{E}$.

To ensure that the model produces a desired sampling distribution, we require the following condition on the weight functions $\lambda(\mathbf{x})$.

Condition 2. For any $\mathbf{x} \in \Theta$, $\lambda(\mathbf{x})$ is continuous in $\mathbf{x}$ and satisfies

i. $\lambda_i(\mathbf{x}) \geqslant 0$ for any $i = 1, \ldots, m$;
ii. $\sum_{i=1}^m \lambda_i(\mathbf{x}) = 1$;
iii. $\lambda_i(\mathbf{x}_j) = 1\{\mathbf{x}_i = \mathbf{x}_j\}$ for all $i, j = 1, \ldots, m$, where $1\{\cdot\}$ is the indicator function.

Notice that in DOvS problems, $\mathbf{x}$ can take only a finite number of values. By requiring $\lambda(\mathbf{x})$ be continuous in Condition 2 we interpret $\lambda(\mathbf{x})$ as a function defined on the real Euclidean space. Indeed, many choices of $\lambda(\mathbf{x})$ satisfy Condition 2. For instance, we may define

$$\lambda_i(\mathbf{x}) = \begin{cases} \dfrac{|1 - \gamma(\mathbf{x}, \mathbf{x}_i)|^{-b}}{\sum_{j=1}^m |1 - \gamma(\mathbf{x}, \mathbf{x}_j)|^{-b}} & \mathbf{x} \neq \mathbf{x}_i \\ 1 & \mathbf{x} = \mathbf{x}_i \end{cases}$$

for some $b > 0$, or

$$\lambda_i(\mathbf{x}) = \begin{cases} \dfrac{\|\mathbf{x} - \mathbf{x}_i\|^{-b}}{\sum_{j=1}^m \|\mathbf{x} - \mathbf{x}_j\|^{-b}} & \mathbf{x} \neq \mathbf{x}_i \\ 1 & \mathbf{x} = \mathbf{x}_i \end{cases}$$

for some $b > 0$, and they both satisfy Condition 2.

Based on the model proposed in Equation (9), we have the following proposition on the mean and variance of $Y(\mathbf{x})$ conditioned on all available information.

Proposition 1. *For any* $\mathbf{x} \in \Theta$, *if Condition* 2 *is satisfied,*

$$\mathrm{E}^*[Y(\mathbf{x})] = \lambda(\mathbf{x})^{\mathrm{T}}\bar{\mathbb{G}},$$

$$\mathrm{Var}^*[Y(\mathbf{x})] = \sigma^2[1 - 2\lambda(\mathbf{x})^{\mathrm{T}}\gamma(\mathbf{x}) + \lambda(\mathbf{x})^{\mathrm{T}}\Gamma\lambda(\mathbf{x})] + \lambda(\mathbf{x})^{\mathrm{T}}\Sigma_{\mathscr{E}}\lambda(\mathbf{x}).$$

Furthermore, $\mathrm{E}^*[Y(\mathbf{x}_i)] = \bar{G}(\mathbf{x}_i)$ *and* $\mathrm{Var}^*[Y(\mathbf{x}_i)] = \sigma^2(\mathbf{x}_i)/n_i$ *for all* $i = 1, \ldots, m$.

Proposition 1 provides an approach to calculating the $\mathrm{E}^*[Y(\mathbf{x})]$ and $\mathrm{Var}^*[Y(\mathbf{x})]$ functions without conducting matrix inversion. Therefore, they can be calculated accurately and efficiently even when Γ and $\Sigma_{\mathscr{E}}$ are ill conditioned. Proposition 1 also shows that the $\mathrm{E}^*[Y(\mathbf{x})]$ and $\mathrm{Var}^*[Y(\mathbf{x})]$ values at any evaluated solution $\mathbf{x}_i$ are its sample mean and the variance of the sample mean. Therefore, we rely solely on the simulation information to predict $g(\mathbf{x})$ if $\mathbf{x}$ has been simulated. Furthermore, when $n_i \to \infty$ or $\sigma^2(\mathbf{x}_i) = 0$, $\mathrm{E}^*[Y(\mathbf{x}_i)] \to g(\mathbf{x}_i)$ and $\mathrm{Var}^*[Y(\mathbf{x}_i)] \to 0$, for any $i = 1, \ldots, m$.

Proposition 1 shows that, for any solution $\mathbf{x}$ that has not been simulated, the $\mathrm{E}^*[Y(\mathbf{x})]$ is a linear combination of $\bar{G}(\mathbf{x}_i)$, $i = 1, \ldots, m$, which are the sample means of the evaluated solutions. Notice that the commonly used weight function $\lambda(\mathbf{x})$, such as those introduced above, depends only on the distances of $\mathbf{x}$ to $\mathbf{x}_1, \ldots, \mathbf{x}_m$, and closer solutions often have higher weights. Then, solutions that are close to the current sample-best point tend to have a high $\mathrm{E}^*[Y(\mathbf{x})]$ value, and solutions that are close to the current sample-worst point tend to have a low $\mathrm{E}^*[Y(\mathbf{x})]$ value. Therefore, the $\mathrm{E}^*[Y(\mathbf{x})]$ function can successfully reflect the necessity of exploitation.

Proposition 1 also shows that, for any solution $\mathbf{x}$ that has not been simulated, the $\mathrm{Var}^*[Y(\mathbf{x})]$ consists of two parts. The first part represents the uncertainty caused by the allocation of the simulated solutions (i.e., $\mathbf{x}_1, \ldots, \mathbf{x}_m$) and the second part represents the uncertainty caused by the estimation error at the simulated solutions. To see how the $\mathrm{Var}^*[Y(\mathbf{x})]$ function reflects the necessity of exploration, we consider the first part of the function. We let

$$\tilde{\sigma}^2(\mathbf{x}) = \sigma^2[1 - 2\lambda(\mathbf{x})^{\mathrm{T}}\gamma(\mathbf{x}) + \lambda(\mathbf{x})^{\mathrm{T}}\Gamma\lambda(\mathbf{x})],$$

which is the first part of $\mathrm{Var}^*[Y(\mathbf{x})]$. A direct analysis of $\tilde{\sigma}^2(\mathbf{x})$ is difficult. However, we can analyze its lower bound provided in the following proposition.

Proposition 2. *Let* $\underline{d}(\mathbf{x}) = \min\{\|\mathbf{x} - \mathbf{x}_1\|, \ldots, \|\mathbf{x} - \mathbf{x}_m\|\}$. *Suppose that Condition* 1 *is satisfied by the correlation function* $h(\cdot)$. *Then,* $\tilde{\sigma}^2(\mathbf{x}) \geqslant \sigma^2[1 - h(\underline{d}(\mathbf{x}))]^2$ *and* $\tilde{\sigma}^2(\mathbf{x}) = 0$ *if* $\underline{d}(\mathbf{x}) = 0$.

By Proposition 2, the lower bound of $\tilde{\sigma}^2(\mathbf{x})$ increases as $\mathbf{x}$ moves away from the set of evaluated solutions and decreases as $\mathbf{x}$ moves closer to one of the evaluated solutions. Therefore, we conclude that the $\mathrm{Var}^*[Y(\mathbf{x})]$ function reflects the necessity of exploration.

From the above analysis we see the model proposed in Equation (9) has the appealing properties and is appropriate in deriving sampling distributions. Then, given the values of $\mathrm{E}^*[Y(\mathbf{x})]$ and $\mathrm{Var}^*[Y(\mathbf{x})]$, we may calculate the $\Pr^*\{Y(\mathbf{x}) > c\}$ where $c = \max\{\bar{G}(\mathbf{x}_1), \ldots, \bar{G}(\mathbf{x}_m)\}$ and derive the sampling distribution according to Equation (8).

3.3. Sampling from the Sampling Distribution

To use the sampling distribution of Equation (8) in each iteration of a random search algorithm, a natural question is how to sample from it effectively and efficiently. Notice that the denominator on the right-hand side of Equation (8) involves all solutions in Θ. When Θ is a large set, as in most practical DOvS problems, calculating the denominator is often impossible. Therefore, our goal is to sample from $f(\mathbf{x})$ without calculating the denominator. In this subsection we propose two sampling algorithms that may achieve this goal.

Notice that $\mathrm{E}^*[Y(\mathbf{x})] = \lambda(\mathbf{x})^{\mathrm{T}}\bar{\mathbb{G}} = \sum_{i=1}^m \lambda_i(\mathbf{x})\bar{G}(\mathbf{x}_i)$ for any $\mathbf{x} \in \Theta$. Because $\lambda_i(\mathbf{x}) \geqslant 0$ and $\sum_{i=1}^m \lambda_i(\mathbf{x}) = 1$, we have $\mathrm{E}^*[Y(\mathbf{x})] \leqslant \max\{\bar{G}(\mathbf{x}_1), \ldots, \bar{G}(\mathbf{x}_m)\} = c$. Because $Y(\mathbf{x})$ is normally distributed, $\Pr^*\{Y(\mathbf{x}) > c\} \leqslant \frac{1}{2}$ for all $\mathbf{x} \in \Theta$. Then,

$$f(\mathbf{x}) = \frac{\Pr^*\{Y(\mathbf{x}) > c\}}{\sum_{\mathbf{z}\in\Theta}\Pr^*\{Y(\mathbf{z}) > c\}} \leqslant \frac{(1/2)|\Theta|}{\sum_{\mathbf{z}\in\Theta}\Pr^*\{Y(\mathbf{z}) > c\}} \cdot \frac{1}{|\Theta|}$$

where $|\Theta|$ denotes the number of points in Θ. Let $u(\mathbf{x}) = 1/|\Theta|$ for all $\mathbf{x} \in \Theta$. Notice that $u(\mathbf{x})$ is the probability mass function of a uniform distribution defined on the set Θ. Let $K = \frac{1}{2}[\sum_{\mathbf{z}\in\Theta}\Pr^*\{Y(\mathbf{z}) > c\}]^{-1}|\Theta|$. Then, K is a constant and $f(\mathbf{x}) \leqslant K \cdot u(\mathbf{x})$ for all $\mathbf{x} \in \Theta$. This motivates us to use an acceptance-rejection algorithm to sample from $f(\mathbf{x})$ (see, for instance, Law and Kelton 2000). The following is the detailed algorithm.

Acceptance-Rejection Sampling (ARS) Algorithm

Step 1. Generate a sample $\mathbf{y}$ uniformly in Θ and U uniformly in $(0, 1)$.

Step 2. If $U \leqslant 2\Pr^*\{Y(\mathbf{y}) > c\}$, accept $\mathbf{y}$ and set $\mathbf{x} = \mathbf{y}$; otherwise, go to Step 1.

The following proposition shows the validity of the ARS algorithm.

Proposition 3. *Suppose* $\mathbf{x}$ *is generated by the ARS algorithm. Then for any* $A \subseteq \Theta$, $\Pr^*\{\mathbf{x} \in A\} = \sum_{\mathbf{y}\in A} f(\mathbf{y})$.

By applying the ARS algorithm, we avoid computing the closed-form expression of $f(\mathbf{x})$ and, thus, significantly improve the efficiency of sampling from $f(\mathbf{x})$. When the probability mass of $f(\mathbf{x})$ is mainly concentrated on small subsets of Θ, however, the acceptance-rejection scheme used in the ARS algorithm may no longer be efficient because the probability of acceptance may be very low. Therefore, we develop an approximate sampling algorithm by using a Markov chain sampling approach (see, for instance, Baumert et al. 2009). The algorithm is as follows.

Markov Chain Coordinate Sampling (MCCS) Algorithm

Step 0. Let $t = 0$, $\mathbf{y} = \mathbf{x}_0$.

Step 1. Let $t = t + 1$. Sample uniformly an integer I from 1 to d. Let $l(\mathbf{y}, I)$ be the line that passes through $\mathbf{y}$ and parallel to the $\mathbf{y}_I$ coordinate axis. Then $l(\mathbf{y}, I)$ intersects with the boundary of Ω (notice that $\Theta \subset \Omega$ and $\Omega \subset \Re^d$

is a convex set) at two points c_1, c_2. Sample an integer j uniformly from $[c_1, \mathbf{y}(I)-1] \cup [\mathbf{y}(I)+1, c_2]$. Set $\mathbf{z} = \mathbf{y}$ and then set $\mathbf{z}(I) = j$.

Step 2. If $U \leqslant f(\mathbf{z})/f(\mathbf{y}) = \Pr^*\{Y(\mathbf{z}) > c\}/\Pr^*\{Y(\mathbf{y}) > c\}$, set $\mathbf{y} = \mathbf{z}$.

Step 3. If $t = T$, return $\mathbf{y}$; otherwise go to Step 1.

Similar to the proof of Baumert et al. (2009), we can show, as $T \to \infty$, the distribution of $\mathbf{y}$ converges to the sampling distribution $f(\cdot)$. The starting solution $\mathbf{x}_0$ of the MCCS algorithm may be the current optimal solution, or any solution in Θ. The MCCS algorithm guarantees to sample (approximately) a solution every T steps. Therefore, it may become more efficient than the ARS algorithm when the acceptance rate in the ARS becomes very low (i.e., lower than $1/T$). Therefore, we may use the MCCS algorithm when the ARS algorithm becomes slow.

4. Gaussian Process-Based Search Algorithm

In this section we introduce a new random search algorithm that incorporates the sampling distribution introduced in §3.2 in each of its iterations. Because the algorithm is guided by a fitted Gaussian process in each iteration, we name it the *Gaussian process-based search* (GPS) algorithm. In this section we provide the detailed GPS algorithm, prove its global convergence, and discuss its stopping criteria.

4.1. GPS Algorithm for DOvS Problems

In the GPS algorithm we let S_k and $\mathbb{S}_k$ denote the sets of simulated solutions in iteration k and through iteration k, respectively, let $n_k(\mathbf{x})$ denote the number of simulation observations through iteration k for all $\mathbf{x} \in \mathbb{S}_k$, and let $\hat{\mathbf{x}}_k^*$ and $\hat{g}_k^*$ denote the sample-best solution and its sample mean through iteration k. In each iteration, say iteration k, we use the solutions in $\mathbb{S}_{k-1}$ and the current sample-best function value $\hat{g}_{k-1}^*$ to construct a sampling distribution $f_k(\mathbf{x})$ and use it to guide the search effort.

We let $\hat{\sigma}_k^2(\mathbf{x})$ denote the sample variance through iteration k for all $\mathbf{x} \in \mathbb{S}_k$. To construct the sampling distribution $f_k(\mathbf{x})$ based on the approach of §3.2, the true variances of $G(\mathbf{x})$, denoted as $\sigma^2(\mathbf{x})$, for all $\mathbf{x} \in \mathbb{S}_{k-1}$ are required in Equation (10). In DOvS problems, however, $\sigma^2(\mathbf{x})$ are typically unknown. In our algorithm we substitute $\sigma^2(\mathbf{x})$ by $\hat{\sigma}_k^2(\mathbf{x})$. However, to prevent the sample variance being zero, we let $\hat{\sigma}_k^2(\mathbf{x}) = \sigma_0^2$ if $\hat{\sigma}_k^2(\mathbf{x}) < \sigma_0^2$, where σ_0^2 is a predetermined small positive constant.[4] We let $\bar{G}_k(\mathbf{x})$ denote the sample mean through iteration k for all $\mathbf{x} \in \mathbb{S}_k$. To avoid the theoretically possible extreme case where $\bar{G}_k(\mathbf{x})$ deviates too much from its true mean $g(\mathbf{x})$ so that $\mathbf{x}$ is not allocated enough sampling effort, we set $\bar{G}_k(\mathbf{x}) = \underline{M}$ if $\bar{G}_k(\mathbf{x}) < \underline{M}$, where $\underline{M}$ is a small constant. If $G(\mathbf{x})$ has a natural lower bound B (e.g., $G(\mathbf{x}) \geqslant 0$), we can just set $\underline{M} = B$; otherwise, we set $\underline{M}$ as an extremely small constant, e.g., $\underline{M} = -10^{10}$.

We also let s denote the number of solutions simulated in each iteration, and r denote the number of simulation observations allocated to each newly simulated solution in each iteration. In the algorithm, both s and r are positive integers specified by the users. The following is the detailed GPS algorithm.

Gaussian Process-Based Search (GPS) Algorithm

Step 0. Let $k = 0$. Sample $\mathbf{x}_{0,1}, \ldots, \mathbf{x}_{0,s}$ independently and uniformly from Θ and let $S_0 = \{\mathbf{x}_{0,1}, \ldots, \mathbf{x}_{0,s}\}$. For every $\mathbf{x} \in S_0$, obtain r independent simulation observations, denoted as $G_{0,1}(\mathbf{x}), \ldots, G_{0,r}(\mathbf{x})$. Let $\mathbb{S}_0 = S_0$. For all $\mathbf{x} \in \mathbb{S}_0$, let $n_0(\mathbf{x}) = r$ and calculate $\bar{G}_0(\mathbf{x})$ and $\hat{\sigma}_0^2(\mathbf{x})$ based on all $n_0(\mathbf{x})$ observations. If $\bar{G}_0(\mathbf{x}) < \underline{M}$, let $\bar{G}_0(\mathbf{x}) = \underline{M}$; and if $\hat{\sigma}_0^2(\mathbf{x}) < \sigma_0^2$, let $\hat{\sigma}_0^2(\mathbf{x}) = \sigma_0^2$. Let $\hat{\mathbf{x}}_0^* = \arg\max_{\mathbf{x} \in \mathbb{S}_0}\{\bar{G}_0(\mathbf{x})\}$ and break the tie arbitrarily if it exists, and let $\hat{g}_0^* = \bar{G}_0(\hat{\mathbf{x}}_0^*)$.

Step 1. Let $k = k+1$. Based on $\mathbf{x}$, $n_{k-1}(\mathbf{x})$, $\bar{G}_{k-1}(\mathbf{x})$ and $\hat{\sigma}_{k-1}^2(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{S}_{k-1}$, construct a sampling distribution

$$f_k(\mathbf{x}) = \frac{\Pr^*\{Y_{k-1}(\mathbf{x}) > \hat{g}_{k-1}^*\}}{\sum_{\mathbf{z} \in \Theta} \Pr^*\{Y_{k-1}(\mathbf{z}) > \hat{g}_{k-1}^*\}}.$$

Step 2. Sample $\mathbf{x}_{k,1}, \ldots, \mathbf{x}_{k,s}$ independently based on $f_k(\mathbf{x})$. Let $S_k = \{\mathbf{x}_{k,1}, \ldots, \mathbf{x}_{k,s}\}$. For every $\mathbf{x} \in S_k$, obtain r independent simulation observations, denoted as $G_{k,1}(\mathbf{x}), \ldots, G_{k,r}(\mathbf{x})$.

Step 3. Let $\mathbb{S}_k = \mathbb{S}_{k-1} \cup S_k$. Let $n_k(\mathbf{x}) = n_{k-1}(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{S}_k \setminus S_k$, $n_k(\mathbf{x}) = n_{k-1}(\mathbf{x}) + r$ for all $\mathbf{x} \in \mathbb{S}_k \cap S_k$, and $n_k(\mathbf{x}) = r$ for all $\mathbf{x} \in S_k \setminus \mathbb{S}_{k-1}$. For all $\mathbf{x} \in \mathbb{S}_k$, calculate (or update) $\bar{G}_k(\mathbf{x})$ and $\hat{\sigma}_k^2(\mathbf{x})$ based on all $n_k(\mathbf{x})$ observations. If $\bar{G}_k(\mathbf{x}) < \underline{M}$, let $\bar{G}_k(\mathbf{x}) = \underline{M}$; and if $\hat{\sigma}_k^2(\mathbf{x}) < \sigma_0^2$, let $\hat{\sigma}_k^2(\mathbf{x}) = \sigma_0^2$.

Step 4. Let $\hat{\mathbf{x}}_k^* = \arg\max_{\mathbf{x} \in \mathbb{S}_k}\{\bar{G}_k(\mathbf{x})\}$ and break the tie arbitrarily if it exists, and let $\hat{g}_k^* = \bar{G}_k(\hat{\mathbf{x}}_k^*)$. Go to *Step* 1.

As pointed out by Hong and Nelson (2007), random search algorithms typically include two schemes in each iteration: a sampling scheme that determines where to sample next solutions and an estimation scheme that determines how to allocate simulation effort to the simulated solutions. In this paper we focus mainly on the sampling scheme by showing how to construct a sampling distribution that balances the exploitation and exploration trade-off. Therefore, to illustrate the usefulness of our idea, we use a very simple estimation scheme in the GPS algorithm, i.e., allocating a fixed number of simulation observations to each of the newly simulated solutions. To make the algorithm practically competent, however, one may consider using other more sophisticated and more efficient estimation schemes.

4.2. Convergence of GPS Algorithm

To prove the global convergence of the GPS algorithm, we first prove the following lemmas.

Lemma 1. *Suppose that the GPS algorithm is used to solve Problem* (1). *For any* $\mathbf{x} \in \Theta$, *if* $f_k(\mathbf{x}) \geqslant c$ *infinitely often* (*i.o.*) *for some constant* $c > 0$, $n_k(\mathbf{x}) \to \infty$ *with probability* 1 (*w.p.*1) *as* $k \to \infty$.

PROOF. It suffices to prove that $\Pr\{\lim_{k\to\infty} n_k(\mathbf{x}) < N\} = 0$ for any $N > 0$ and for any $\mathbf{x}$ such that $f_k(\mathbf{x}) \geqslant c$ i.o. Notice that, if $f_k(\mathbf{x}) \geqslant c$,

$$\Pr\{\mathbf{x} \in S_k\} = 1 - \Pr\{\mathbf{x} \notin S_k\} = 1 - [1 - f_k(\mathbf{x})]^s \geqslant 1 - (1-c)^s > 0.$$

Then,

$$\Pr\left\{\lim_{k\to\infty} n_k(\mathbf{x}) < N\right\} = \Pr\left\{\lim_{k\to\infty} r \sum_{i=1}^{k} 1_{\{\mathbf{x}\in S_i\}} < N\right\} \leqslant \Pr\left\{\lim_{k\to\infty} r \sum_{i\leqslant k,\, f_i(\mathbf{x})\geqslant c} 1_{\{\mathbf{x}\in S_i\}} < N\right\}.$$

Let $n_k = \sum_{i=1}^{k} 1_{\{f_i(\mathbf{x})\geqslant c\}}$ and B_i, $i = 1, 2, \ldots$, be independent Bernoulli random variables with parameter $1 - (1-c)^s$. Then,

$$\Pr\left\{\lim_{k\to\infty} n_k(\mathbf{x}) < N\right\} \leqslant \Pr\left\{\lim_{k\to\infty} r \sum_{j=1}^{n_k} B_j < N\right\} = \Pr\left\{\lim_{k\to\infty} \frac{1}{n_k} \sum_{j=1}^{n_k} B_j < \frac{N}{n_k r}\right\}. \tag{11}$$

Because $f_k(\mathbf{x}) \geqslant c$ i.o., we have $n_k \to \infty$ as $k \to \infty$. Then, by the strong law of large numbers,

$$\frac{1}{n_k} \sum_{j=1}^{n_k} B_j \to 1 - (1-c)^s \quad \text{w.p.1}$$

as $k \to \infty$. Furthermore, notice that $N/(n_k r) \to 0$ w.p.1. Then, we have that the right-hand side of Equation (11) equals to zero. Therefore, $n_k(\mathbf{x}) \to \infty$ w.p.1 as $k \to \infty$. □

LEMMA 2. *Suppose that the GPS algorithm is used to solve Problem* (1). *Then,* $n_k(\hat{\mathbf{x}}_k^*) \to \infty$ *w.p.*1 *as* $k \to \infty$.

PROOF. Because Θ is a finite set, the sequence $\hat{\mathbf{x}}_k^*$ has a finite set of cluster points $\mathscr{C} \subset \Theta$ such that $\hat{\mathbf{x}}_k^* = \mathbf{x}$ i.o. for all $\mathbf{x} \in \mathscr{C}$ and

$$\Pr\{\hat{\mathbf{x}}_k^* \notin \mathscr{C} \text{ i.o.}\} = 0. \tag{12}$$

Notice that, where $\hat{\mathbf{x}}_{k-1}^* = \mathbf{x}$,

$$f_k(\mathbf{x}) = \frac{1/2}{\sum_{\mathbf{x}\in\Theta} \Pr^*\{Y_{k-1}(\mathbf{z}) > \hat{g}_{k-1}^*\}} \geqslant \frac{1}{|\Theta|}.$$

Then, by Lemma 1,

$$\Pr\{n_k(\mathbf{x}) \to \infty \text{ as } k \to \infty \text{ for all } \mathbf{x} \in \mathscr{C}\} = 1. \tag{13}$$

Combining Equations (12) and (13), it is easy to see that $n_k(\hat{\mathbf{x}}_k^*) \to \infty$ w.p.1 as $k \to \infty$. □

LEMMA 3. *Suppose that the GPS algorithm is used to solve Problem* (1). *Then, for any* $\epsilon > 0$, $\limsup_{k\to\infty} \hat{g}_k^* < g^* + \epsilon$ *w.p.*1.

PROOF. Notice that it is equivalent to prove that $\Pr\{\hat{g}_k^* > g^* + \epsilon \text{ i.o.}\} = 0$, where $g^* = \max_{\mathbf{x}\in\Theta} g(\mathbf{x})$.

To facilitate the proof, suppose at the end of each iteration, for any $\mathbf{x} \in \Theta$, we add additional observations to $\mathbf{x}$ such that the number of observations at $\mathbf{x}$ is at least $n_k(\hat{\mathbf{x}}_k^*)$. Let $N_k(\mathbf{x}) = \max\{n_k(\hat{\mathbf{x}}_k^*), n_k(\mathbf{x})\}$ and let

$$\tilde{G}_k(\mathbf{x}) = \frac{1}{N_k(\mathbf{x})} \sum_{i=1}^{N_k(\mathbf{x})} G_i(\mathbf{x}).$$

By Lemma 2, $N_k(\mathbf{x}) \to \infty$ w.p.1 as $k \to \infty$. Then, by the strong law of large numbers, $\tilde{G}_k(\mathbf{x}) \to g(\mathbf{x})$ w.p.1 as $k \to \infty$. Then,

$$\Pr\{\hat{g}_k^* > g^* + \epsilon \text{ i.o.}\} \leqslant \Pr\left\{\max_{\mathbf{x}\in\Theta} \tilde{G}_k(\mathbf{x}) > g^* + \epsilon \text{ i.o.}\right\} \leqslant \sum_{\mathbf{x}\in\Theta} \Pr\{\tilde{G}_k(\mathbf{x}) > g^* + \epsilon \text{ i.o.}\} = 0,$$

where the last equality holds because $g^* = \max_{\mathbf{x}\in\Theta} g(\mathbf{x}) \geqslant g(\mathbf{x})$ for all $\mathbf{x} \in \Theta$ and $\tilde{G}_k(\mathbf{x}) \to g(\mathbf{x})$ w.p.1 as $k \to \infty$ for all $\mathbf{x} \in \Theta$. □

LEMMA 4. *Suppose that the GPS algorithm is used to solve Problem* (1) *and Conditions* 1 *and* 2 *are satisfied. Then,* $n_k(\mathbf{x}) \to \infty$ *w.p.*1 *for all* $\mathbf{x} \in \Theta$.

PROOF. It suffices to prove that $\Pr\{\lim_{k\to\infty} n_k(\mathbf{x}) < N\} = 0$ for any $\mathbf{x} \in \Theta$ and any $N > 0$. By Lemma 3, for all $\mathbf{x} \in \Theta$,

$$\Pr\left\{\lim_{k\to\infty} n_k(\mathbf{x}) < N\right\} = \Pr\left\{\lim_{k\to\infty} n_k(\mathbf{x}) < N, \limsup_{k\to\infty} \hat{g}_k^* < g^* + \epsilon\right\}$$

for any $\epsilon > 0$. Let $\Omega(\mathbf{x}) = \{\lim_{k\to\infty} n_k(\mathbf{x}) < N, \limsup_{k\to\infty} \hat{g}_k^* < g^* + \epsilon\}$. It suffices to show that $\Pr\{\Omega(\mathbf{x})\} = 0$ for all $\mathbf{x} \in \Theta$.

For any $\mathbf{x} \in \Theta$ and $\omega \in \Omega(\mathbf{x})$, there exists $K(\omega)$ such that for any $k > K(\omega)$, $\hat{g}_k^*(\omega) < g^* + \epsilon$. For any $k > K(\omega)$, if no observation has been allocated to $\mathbf{x}$ by iteration k, i.e., $n_k(\mathbf{x}) = 0$, by Proposition 2, we have

$$\mathrm{Var}^*[Y_k(\mathbf{x})] \geqslant \tilde{\sigma}^2(\mathbf{x}) \geqslant \sigma^2(1 - h(\underline{d}(\mathbf{x})))^2 \geqslant \underline{\sigma}^2,$$

where $\underline{\sigma}^2 = \sigma^2(1 - h(\underline{d}))$ with $\underline{d} = \min\{\|\mathbf{x} - \mathbf{y}\|, \mathbf{x} \neq \mathbf{y}, \mathbf{x}, \mathbf{y} \in \Theta\} > 0$. If by iteration k there have been some observations allocated to $\mathbf{x}$, i.e., $n_k(\mathbf{x}) > 0$, Proposition 1 suggests that

$$\mathrm{Var}^*[Y_k(\mathbf{x})] = \frac{\max\{\hat{\sigma}(\mathbf{x})^2, \sigma_0^2\}}{n_k(\mathbf{x})} \geqslant \frac{\sigma_0^2}{N}.$$

Let $\tilde{\sigma}^2 = \min\{\underline{\sigma}^2, \sigma_0^2/N\}$. By the above analysis, we have $\mathrm{Var}^*[Y_k(\mathbf{x})] \geqslant \tilde{\sigma}^2$.

Furthermore, we have $\mathrm{E}^*[Y_k(\mathbf{x})] \geqslant \underline{M}$ for any $\mathbf{x} \in \Theta$ and $k > 0$. Then, for any $k > K(\omega)$,

$$\Pr^*\{Y_k(\mathbf{x}) > \hat{g}_k^*\} \geqslant \Phi\left(\frac{g^* + \epsilon - \underline{M}}{\tilde{\sigma}}\right) > 0,$$

where $\Phi(\cdot)$ is the cumulative distribution function of a standard normal random variable. Then, we have

$$f_k(\mathbf{x}) \geqslant \frac{1}{|\Theta|}\Phi\left(\frac{g^* + \epsilon - \underline{M}}{\tilde{\sigma}}\right),$$

where the right-hand side of the inequality is a positive constant. Then, by Lemma 1, we have $\Pr\{\Omega(\mathbf{x})\} = 0$ for all $\mathbf{x} \in \Theta$. □

Now we can present the main result on the convergence of the GPS algorithm.

THEOREM 1. *Suppose that the GPS algorithm is used to solve Problem* (1) *and Conditions* 1 *and* 2 *are satisfied. Then,* $\hat{g}_k^* \to g^*$ *w.p.*1 *as* $k \to \infty$.

PROOF. Notice that

$$|\hat{g}_k^* - g^*| = \left|\max_{x\in\Theta} \bar{G}_k(\mathbf{x}) - \max_{\mathbf{x}\in\Theta} g(\mathbf{x})\right| \leqslant \max_{\mathbf{x}\in\Theta} |\bar{G}_k(\mathbf{x}) - g(\mathbf{x})|.$$

Therefore, for any $\epsilon > 0$,

$$\begin{aligned}\Pr\{|\hat{g}_k^* - g^*| > \epsilon \text{ i.o.}\} &\leqslant \Pr\left\{\max_{\mathbf{x}\in\Theta} |\bar{G}_k(\mathbf{x}) - g(\mathbf{x})| > \epsilon \text{ i.o.}\right\} \\ &\leqslant \sum_{\mathbf{x}\in\Theta} \Pr\{|\bar{G}_k(\mathbf{x}) - g(\mathbf{x})| > \epsilon \text{ i.o.}\} = 0,\end{aligned}$$

where the last equation follows from Lemma 4, the strong law of large numbers, and the fact that Θ is a finite set. Therefore, we conclude that $\hat{g}_k^* \to g^*$ w.p.1 as $k \to \infty$. □

4.3. Stopping Criteria

Theorem 1 shows that the sequence of sample-best solutions generated by the GPS algorithm converges to the global optimum as the simulation effort goes to infinity. In practice, however, the simulation effort is limited and the algorithm has to stop short of infinity. Then, one natural question is when to stop. A related problem is the infinite-time inference problem, which infers the optimality gap of a DOvS problem based on the finite-time information. This problem was first proposed by Andradóttir and Nelson (2002), but since then there has been very little progress in solving it.

A few stopping criteria have been proposed in the literature and used in practice. The most widely used stopping criterion for DOvS algorithms is to stop the algorithm when the available computation time is exhausted. However, this stopping criterion is often too passive because the users may not know exactly how much computation time is available to them and the criterion does not provide any information on the quality of the solution at the stopping. Xu et al. (2010) proposed a hypothesis test based stopping criteria for locally convergent DOvS algorithms, which tests whether the current solution is the best in its local neighborhood after all solutions in the neighborhood being simulated. However, this stopping criterion cannot be extended to globally convergent DOvS algorithms, because the global optimality requires the solution be the best among all feasible solutions and the algorithms often do not have the computation budget available to simulate all feasible solutions.

Indeed, proposing stopping criteria for a globally convergent DOvS algorithm is in general very difficult. When there is no structural information on the objective function $g(\mathbf{x})$, the global optimality of a solution cannot be established without simulating all feasible solutions. In practice, however, we typically cannot afford to simulate all solutions before stopping. Therefore, we have to impose certain structures on the objective function $g(\mathbf{x})$ to develop reasonable stopping criteria.

In this subsection we propose to use the Gaussian process model of §3 to describe the structure of $g(\mathbf{x})$ and use the derived distribution of $Y(\mathbf{x})$ to design meaningful stopping criteria. From an informal Bayesian viewpoint, $Y(\mathbf{x})$ is our posterior model of $g(\mathbf{x})$ and, therefore, we may use it to infer the optimality gap in a probabilistic sense. For instance, we may consider the following four statistics at the end of iteration k,

$$\begin{aligned}\Delta_{k,1} &= \frac{1}{|\Theta|}\sum_{\mathbf{x}\in\Theta} \Pr^*\{Y_k(\mathbf{x}) \geqslant \hat{g}_k^*\},\\ \Delta_{k,2} &= \frac{1}{|\Theta|}\sum_{\mathbf{x}\in\Theta} \mathrm{E}^*[(Y_k(\mathbf{x}) - \hat{g}_k^*)^+],\\ \Delta_{k,3} &= \sum_{\mathbf{x}\in\Theta} \Pr^*\{Y_k(\mathbf{x}) \geqslant \hat{g}_k^*\} f_{k+1}(\mathbf{x}),\\ \Delta_{k,4} &= \sum_{\mathbf{x}\in\Theta} \mathrm{E}^*[(Y_k(\mathbf{x}) - \hat{g}_k^*)^+] f_{k+1}(\mathbf{x}).\end{aligned}$$

Notice that $\Delta_{k,1} = \Pr^*\{Y_k(\mathbf{U}) \geqslant \hat{g}_k^*\}$ and $\Delta_{k,2} = \mathrm{E}^*[(Y_k(\mathbf{U}) - \hat{g}_k^*)^+]$, where $\mathbf{U}$ is a random vector that is uniformly distributed on Θ and independent of other randomness. Then, $\Delta_{k,1}$ is the expected conditional probability that a uniformly generated solution on Θ has a value that is greater than the current sample best solution and $\Delta_{k,2}$ is the expected conditional improvement. Therefore, we may stop the algorithm when the values of $\Delta_{k,1}$ and $\Delta_{k,2}$ are small enough, meaning that the chance of finding a better solution is small enough so that we may not have a significant loss if we stop the algorithm at iteration k.

Similarly, $\Delta_{k,3} = \Pr^*\{Y_k(\mathbf{X}) \geqslant \hat{g}_k^*\}$ and $\Delta_{k,4} = \mathrm{E}^*[(Y_k(\mathbf{X}) - \hat{g}_k^*)^+]$, where $\mathbf{X}$ is a random vector that is distributed according to $f_{k+1}(\mathbf{x})$ and independent of other randomness. Then, $\Delta_{k,3}$ is the expected conditional probability that a better solution can be found in iteration $k+1$ and $\Delta_{k,4}$ is the expected conditional improvement in iteration $k+1$. Therefore, we may stop the algorithm when the values of $\Delta_{k,3}$ and $\Delta_{k,4}$ are small enough, indicating that the gain from an additional iteration is small enough.

Although the exact calculations of $\Delta_{k,1}$ to $\Delta_{k,4}$ may not be possible as they require evaluations of all $\mathbf{x} \in \Theta$, these quantities can be estimated using a finite sample of $\mathbf{U}$ and $\mathbf{X}$. Furthermore, if the stopping criteria are not satisfied at

iteration k, the samples are not wasted as they can be used in *Step* 2 of the GPS algorithm in iteration $k+1$.

The discussions on the stopping criteria in this subsection are quite conceptual. Nevertheless, we believe that they provide a direction to solve this important yet significantly understudied problem. In §5 we conduct some numerical evaluations of these stopping criteria.

It is worthwhile to notice that there are other types of stopping criteria for global optimization in the context of Bayesian methods that are similar to our stopping criteria. For instance, Betrò (1991) proposed the *k-sla* stopping rules, which call for stopping the sampling process as soon as the current cost is not greater than the cost expected if at most k further observations are taken. Readers may refer to Betrò (1991) for a more detailed discussion on the rule and Sasena (2002) for the literature review of some other related ideas.

5. Numerical Examples

In this section we study the performance of the GPS algorithm and the stopping criteria, analyze the effect of parameter settings on the performance of the GPS algorithm, and compare the GPS algorithm with other competing algorithms, through two numerical examples.

5.1. A Problem with Multiple Local Optima

The first problem that we want to test is as follows:

$$\max_{0\leqslant x_1,\,x_2\leqslant 100} g(x_1,x_2) = 10\cdot\frac{\sin^6(0.05\pi x_1)}{2^{2((x_1-90)/50)^2}} + 10\cdot\frac{\sin^6(0.05\pi x_2)}{2^{2((x_2-90)/50)^2}}. \tag{14}$$

A similar example is used by Xu et al. (2010). Notice that g has 25 local optima with the global optimum $g(90,90) = 20$ and the second best local optimum $g(90,70) = g(70,90) = 18.95$ (see Figure 5). Without knowing the closed-form expression of g, this problem is difficult to solve because it has many local optima and the difference between the largest and the second largest function values is quite small.

To make this problem a DOvS problem, we discretize the solution set to $\Theta = \{(x_1,x_2) \mid x_1 = 0.01z_1, x_2 = 0.01z_2, z_1, z_2 = 1,\ldots,10{,}000\}$ and add a noise term that is normally distributed with mean 0 and variance 1 to all solutions in Θ. For the GPS algorithm, we set $\sigma = 4$, $\gamma_i(\mathbf{x}) = \exp\{-\|\mathbf{x}-\mathbf{x}_i\|^{0.5}\}$, $\lambda_i(\mathbf{x}) = \|\mathbf{x}-\mathbf{x}_i\|^{-4}/(\sum_{i=1}^{n}\|\mathbf{x}-\mathbf{x}_i\|^{-4})$, $s = 5$ and $r = 10$, and apply the ARS algorithm to sample solutions in each iteration.

The left panel of the Figure 6 shows the 30 sample paths of the GPS algorithm with a total sample size of 10^4 (approximately 1,000 evaluated solutions). It shows that, in all 30 runs, the GPS algorithm can find sample-best solutions that are very close to the true optimal solution. With less than 500 evaluated points, the GPS algorithm is able to identify the optimal region. To compare the GPS algorithm with other algorithms, we also apply a simulated annealing algorithm and a global random search (GRS) algorithm to solve this problem. In particular, we use the simulated annealing algorithm of Alrefaei and Andradóttir (1999) with the constant temperature being 0.1, the local neighborhood being a hypercube with length 5 centered at the current solution, and the number of observations being 5 for each visit of a solution, and the GRS algorithm of Andradóttir (1996) with the number of observations being 5 for each visit of a solution. The right panel of Figure 6 shows the average performances of all three algorithms over 30 sample paths, from which we see that the performance of the GPS algorithm is significantly better than the performances of the other two algorithms with the same total sample size. To further illustrate the performance of the GPS algorithm, in Figure 7 we plot the evaluated points, the mean surface (i.e., $\mathrm{E}^*[Y_k(\mathbf{x})]$), the variance surface (i.e., $\mathrm{Var}^*[Y_k(\mathbf{x})]$) and the probability surface (i.e., $\Pr^*\{Y_k(\mathbf{x}) > \hat{g}_k^*\}$) through different iterations in a typical run. From these figures we see that the GPS algorithm at the beginning mainly conducts explorative search to find regions with good performance and then switches to exploitative search. As the number of iterations increases, the fitted surface gets closer to the true surface, and the sampling distribution focuses more on the optimal region. This demonstrates the strength of the GPS algorithm in balancing the exploitation and exploration trade-off.

Notice that the simulated annealing algorithm is an example of exploitation-based algorithms while the GRS algorithm is an example of exploration-based algorithms. Exploitation-based algorithms are not suitable for solving problems with many local optima, because they are easy to be trapped to a local optimum. Exploration-based algorithms are more suitable because they can "jump" out of a locally optimal region. To further show the strength of the GPS algorithm, based on 1,000 runs of each algorithm, we plot the distributions of current sample-best solutions of the

Figure 5. (Color online) The $g(x_1,x_2)$ function defined in Equation (14).

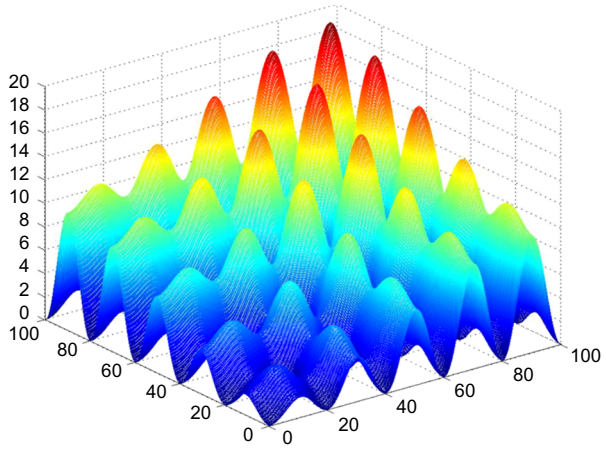


Figure 6. Estimated optimal value for function g.

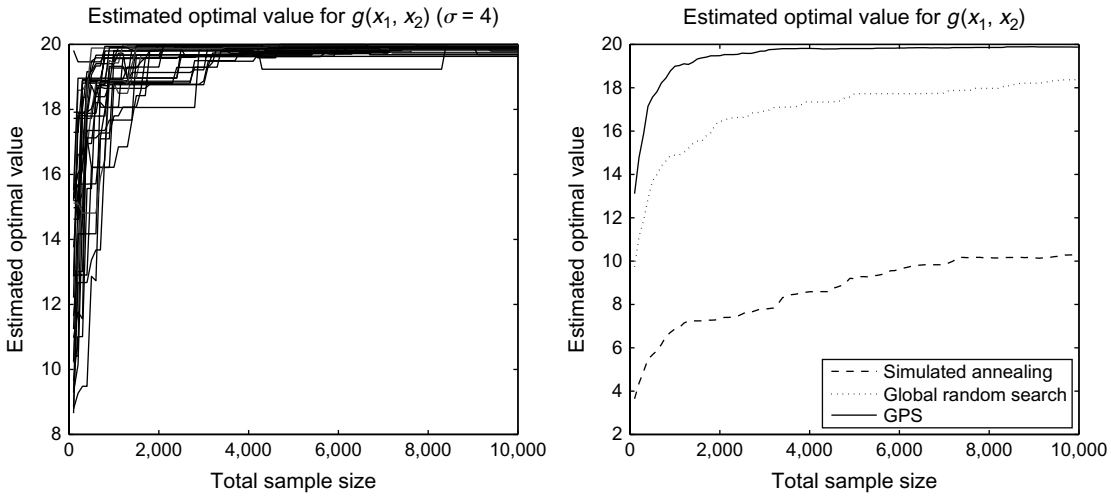
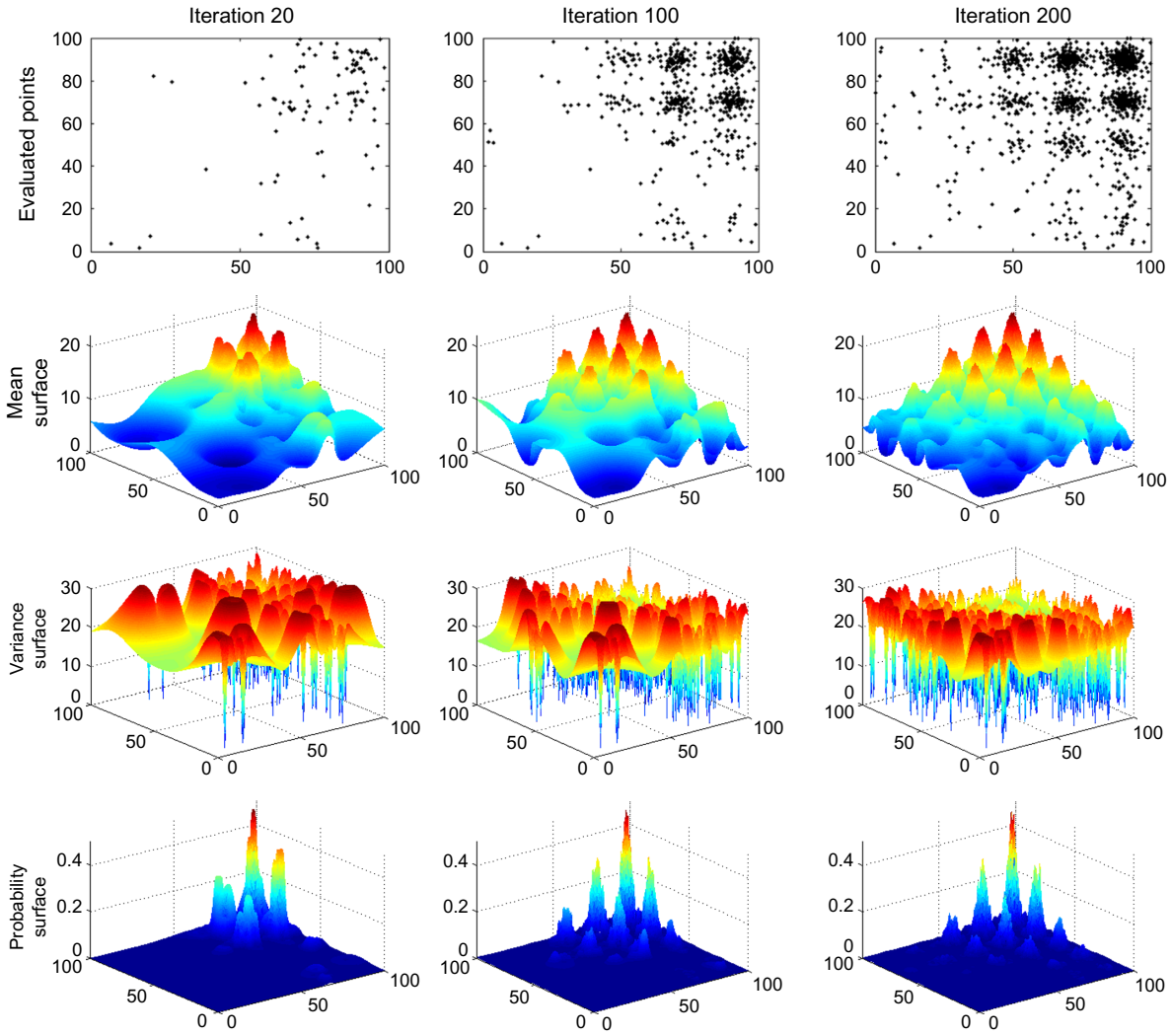


Figure 7. (Color online) The evaluated points, mean surface, variance surface, and probability surface by 20, 100, 200 iterations (approximately 100, 500, and 1,000 evaluated points) when solving Problem (14).



GPS and GRS algorithms after 1,000, 2,500, 5,000 observations in Figure 8. From the figure we can see that the GPS algorithm identifies good regions faster than the GRS algorithm does and is capable of conducting exploitative search while the GRS algorithm is not.

5.2. An Inventory Problem

The second problem we want to test is an inventory management problem for an assemble-to-order system. We use the same parameter settings and simulation scheme as in Hong and Nelson (2006). This problem is a DOvS problem with eight decision variables and 2.56×10^{10} feasible solutions. Based on the results reported by Hong and Nelson (2006), the problem appears to have multiple local optima that are close to each other and have similar performances, and thus it is particularly suitable for locally convergent random search algorithms like the COMPASS algorithm and exploitation-based algorithms like the simulated annealing algorithms.

The settings of the GPS algorithm for this problem are the same as those for the problem in §5.1, except that we set $\sigma = 15$. (We discuss more on the setting of σ in §5.3.) To speed up the sampling process, we use the MCCS algorithm with $T = 1{,}000$ and the current sample-best solution as the starting point to generate points from the sampling distribution in each iteration of the GPS algorithm. The left panel of Figure 9 shows the 30 sample paths of the GPS algorithm with a total sample size of 1.5×10^4. It shows that, in all 30 runs, the GPS algorithm can find sample-best solutions with very good performances very quickly. We also compare the GPS algorithm with the simulated annealing algorithm, the GRS algorithm and the COMPASS algorithm on this problem. The settings of the GRS algorithm and the simulated annealing algorithm are the same as those in §5.1 except that, for the simulated annealing algorithm, the temperature is set to 1 and the neighborhood is set to be a hypercube with length 3, and the setting of the COMPASS algorithm is the same as those of Hong and Nelson (2006). The right panel of Figure 9 shows the average performances of all four algorithms over 30 sample paths. From this figure, we can see that the performance of the GPS algorithm is only slightly worse than that of the COMPASS algorithm, similar to that of the simulated annealing algorithm, and clearly better than that of the GRS algorithm. Because the number of observations allocated to the current optimal solutions is much smaller in the COMPASS algorithm than in the GPS algorithm, the estimated sample-best objective values of the COMPASS algorithm have often higher biases than those of the GPS algorithm. By adding more observations to the current sample-best solutions to remove the random noise, we find that the actual performances of the GPS and the COMPASS algorithm are nearly the same. Considering that this inventory problem itself is particularly suitable for the COMPASS algorithm that does not have global convergence, the performance of the GPS algorithm is quite satisfactory.

5.3. The Effect of σ^2 on the Performance of GPS

In the GPS algorithm the variance (i.e., σ^2) of the unconditional Gaussian process $M(\mathbf{x})$, is an important parameter. It plays an important role in balancing the exploitation and exploration trade-off. When σ^2 is large, the variance of the Gaussian process is high and the GPS algorithm spends more effort in exploration. When σ^2 is small, the variance of the Gaussian process is low and the GPS algorithm spends more effort in exploitation. To test the impact of σ^2 on the performance of the GPS algorithm, we run the algorithm 30 times with $\sigma = 3, 4, 5$ for Problem (14) and with $\sigma = 10, 15, 20$ for the inventory problem.

In Figure 10, we plot the 30 sample paths as well as the evaluated points through different iterations in a typical run of the GPS algorithm with different values of σ for Problem (14). In Figure 11, we plot the 30 sample paths with different values of σ for the inventory problem. From these figures, we can see that when σ is small (e.g., $\sigma = 3$ and $\sigma = 10$), the algorithm tends to spend more effort in exploitation; and when σ is large (e.g., $\sigma = 5$ and $\sigma = 20$), the algorithm tends to spend more effort in exploration. Therefore, based on the information about the problem in hand, the users of the GPS algorithm may adjust the exploitation and exploration trade-off by adjusting the value of σ.

In practical applications, if the computation budget is limited compared to the size of the solution set, a small σ may be applied in the GPS algorithm to achieve satisfactory finite time performance with a high probability of finding a local optimum. If the computation budget is high, a large σ may be appropriate to increase the probability of finding the global optimum. If the computation budget is not a critical concern, the value of σ may change with the search process of the algorithm. For instance, if after some iterations of search, the function seems to have multiple local optima with similar function values, it is more appropriate to set σ as a relatively large value to avoid being trapped to a local optimum. If solutions with good function values seem to cluster in a small region, then it is more appropriate to set σ as a relatively small value to focus on exploitative search. How to design a scheme that can change σ^2 adaptively is certainly an interesting topic for future study.

For the GPS algorithm, we also need to set some other parameters such as $\lambda(\cdot)$, $\gamma(\cdot)$, and so on. We find that the performance of the GPS algorithm is not sensitive to these parameters.

5.4. Computational Overhead and Efficiency of Random Search Algorithms

Random search algorithms are often compared based on the quality of the solutions when the algorithms use the same number of observations, as we have done in §§5.1 and 5.2. The basic rationales for this type of comparison should be twofold. First, simulation observations typically take a significantly larger amount of computation time than

Figure 8. Distributions of current sample-best solutions after 1,000, 2,500, and 5,000 observations of the GPS and GRS algorithms when solving Problem (14) based on 2,000 runs of each algorithm.

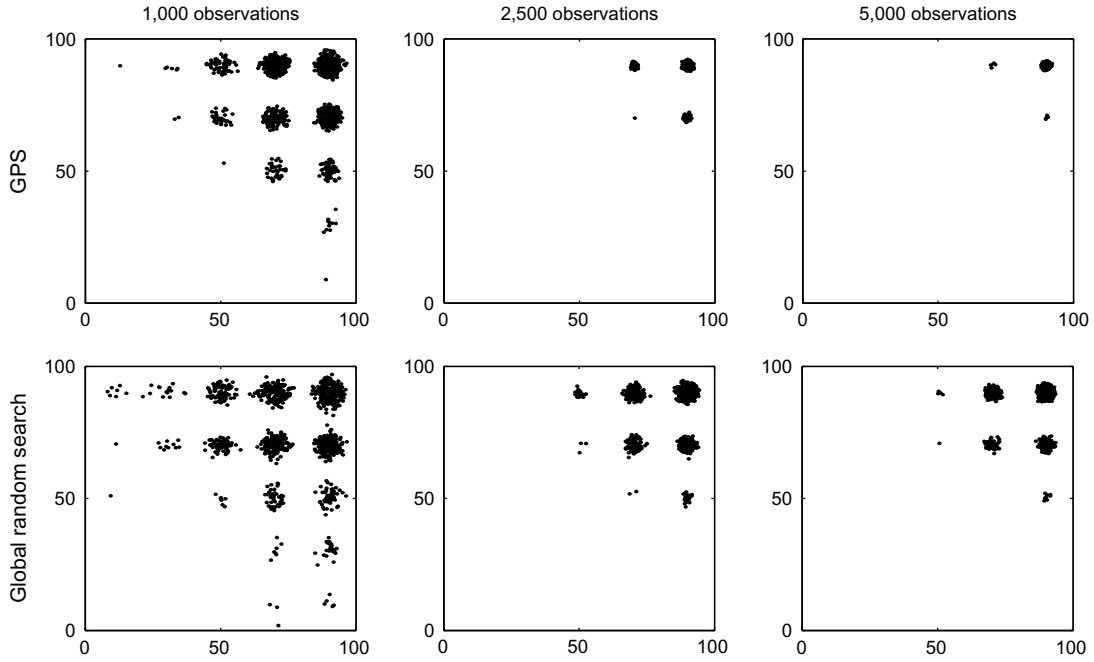
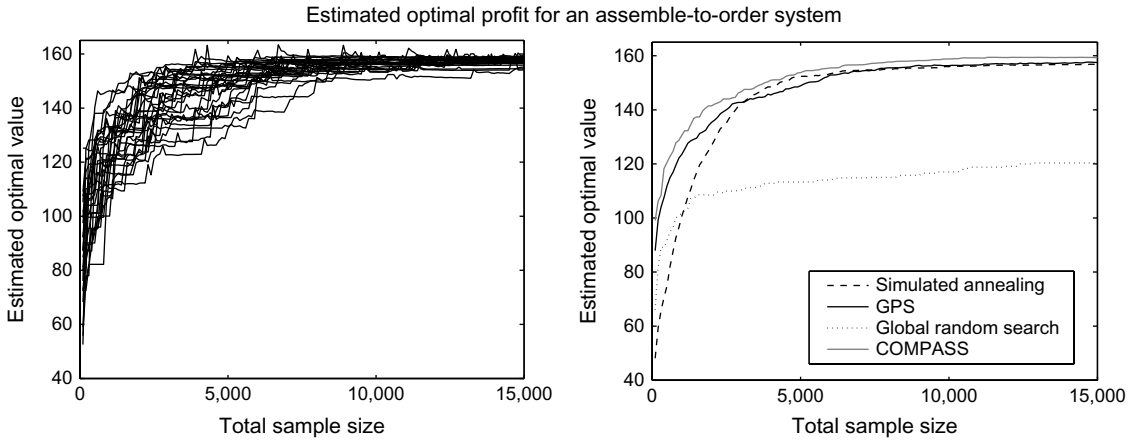


Figure 9. Estimated optimal profit for the assemble-to-order system.



the overhead of the algorithms (e.g., the computation time used in sampling). Therefore, the computational overhead may be ignored in the comparisons. Second, test problems such as Problem (14) often involve only the evaluation of a closed-form expression and require significantly less computation time than a practical simulation experiment. Therefore, actual computation time of algorithms may not be fairly compared. In this subsection we argue that the computational overhead is an important factor to consider when selecting a random search algorithm. If the simulation experiments are fast, algorithms that need more observations but significantly less overhead may outperform algorithms that need less observations but significantly more overhead.

To illustrate our ideas we consider three algorithms: the GRS algorithm that has very little computational overhead, the sequential kriging optimization (SKO) algorithm of Huang et al. (2006) that requires a significant amount of overhead, and the GPS algorithm that is in between. The SKO algorithm is an extension of the EGO algorithm to stochastic problem with a key ingredient called "augmented expected improvement" (AEI) in the method. In each iteration, the algorithm fits a Gaussian process and chooses the solution with the largest AEI value as the next solution for evaluation by solving a global optimization problem, therefore requiring a significantly large amount of computational overhead.[5] We again consider Problem (14) and the inventory problem. The first problem requires very little computation time to evaluate a solution, while the second problem requires some time but not too significant an amount. We compare the performances of the three

Figure 10. Performance and simulated points allocation of the GPS algorithm with $\sigma = 3, 4, 5$ for solving Problem (14).

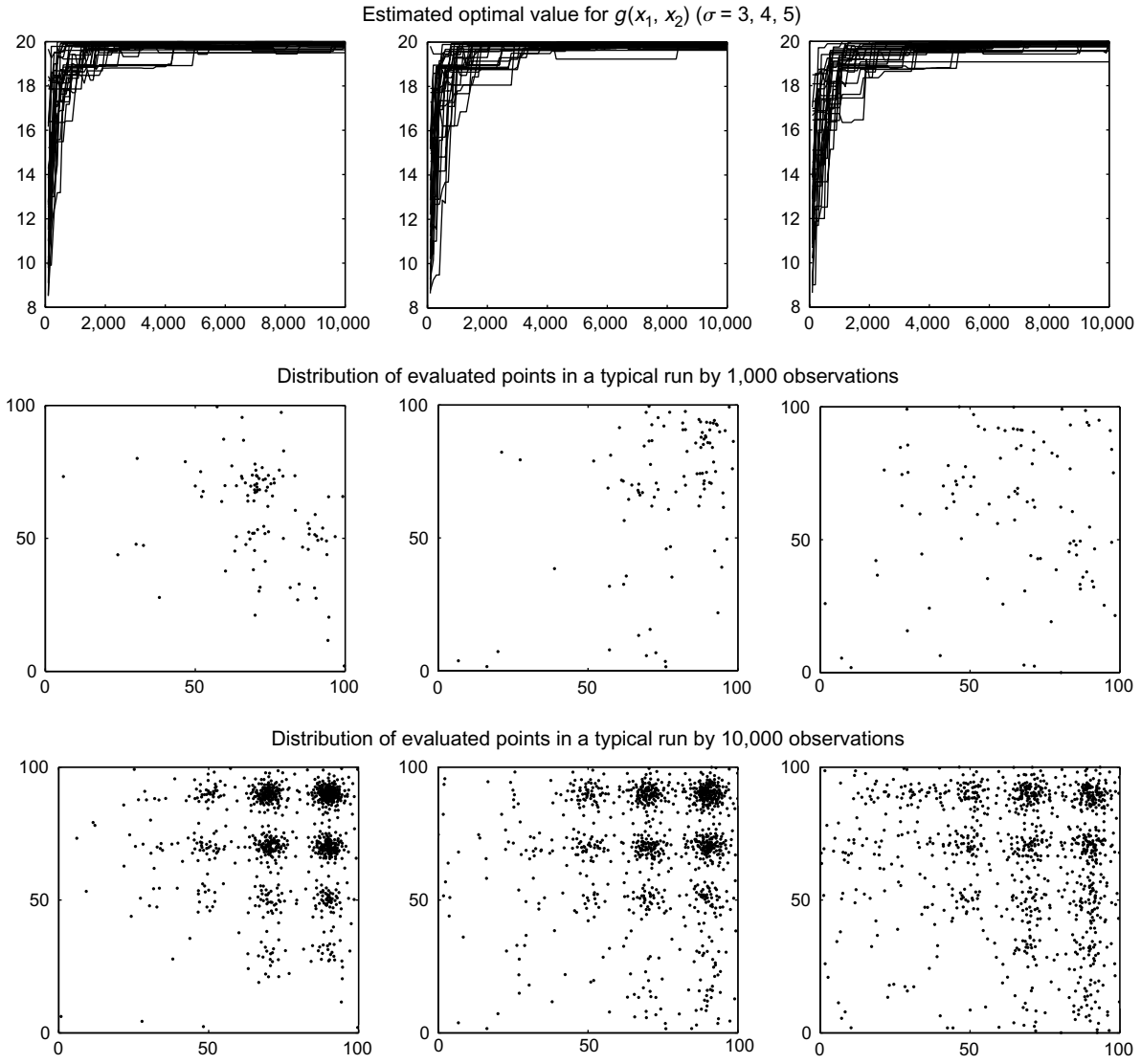
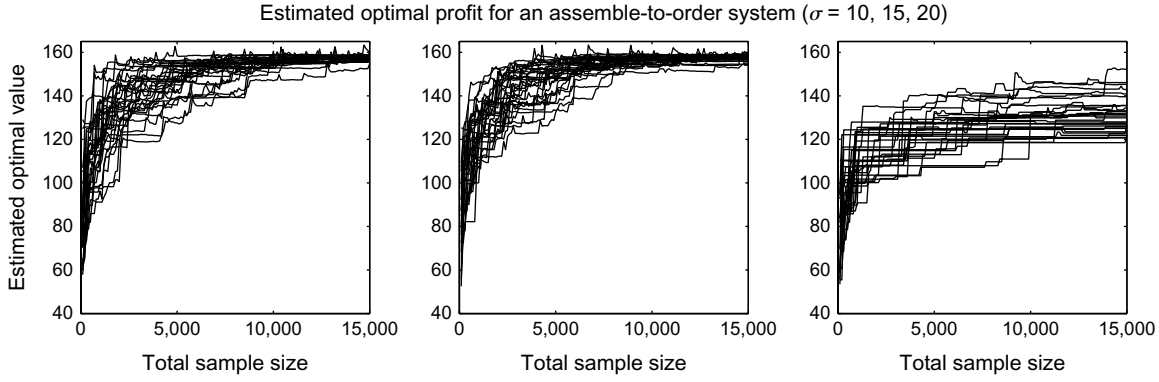


Figure 11. Performance of the GPS algorithm with $\sigma = 10, 15, 20$ for the inventory problem.



algorithms for the two test problems in terms of both the total number of observations and total computation time, and plot 30 sample paths in Figures 12 and 13.

From the upper four plots of Figure 12 we find that, when solving Problem (14), the GPS algorithm has a better performance than the GRS algorithm in terms of the total sample size (with a maximum of 8,000 observations), but a worse performance in terms of the total computation time (with a maximum of 10 seconds). From the bottom four plots of Figure 12 (note that the scales of the bottom four plots are different from those of the upper four) we find that, when solving the same problem, the SKO algorithm has a better performance than the GPS algorithm in terms of the total sample size (with a maximum of 1,000 observations because the SKO algorithm cannot afford to run 8,000 observations), but a much worse performance in terms of the total computation time (with a maximum of 200 seconds). Because Problem (14) requires very little simulation time, nearly all computation time is spent on the algorithm overhead for all three algorithms. From these results we find that the SKO algorithm has a much more significant overhead than the GPS algorithm, and the GPS algorithm has a much more significant overhead than the GRS algorithm. For problems like Problem (14), the GRS is clearly the most efficient of the three even though it requires the largest number of observations to solve the problem.

However, Problem (14) is not a good representative of practical DOvS problems that we are interested in because it requires too little time for simulation. Instead, the inventory problem is a better representative. From the eight plots of Figure 13 we find that, when solving the problem, the GPS algorithm has a better performance than the GRS algorithm in terms of both the total sample size and the total computation time, and a better performance than the SKO algorithm in terms of the total computation time but a worse performance in terms of the total sample size. Therefore, for problems like the inventory problem, the GPS algorithm is clearly the most efficient of the three and should be used in practice. It is worthwhile noting that the SKO algorithm, as well as the *P*-algorithm and the EGO algorithm, are not designed to solve the type of problems that we are interested in, where we may afford thousands to even millions of simulation observations. They are designed to solve problems where function evaluations are very time consuming. For those problems, the overhead of these algorithms may be negligible when compared to the time of functional evaluations, and they should be preferred to the GPS and GRS algorithms.

Remark 2. Although the computational cost for generating a sample point is much lower in the GPS algorithm than in kriging-based algorithms such as the SKO, the computational cost in GPS for generating one sample point from the sampling distribution increases with the number of simulated points. For high-dimensional problems with a large solution set, though in theory we can prove that it is guaranteed to find the global optimum as the computation budget goes to infinity if one applies the GPS algorithm, the finite-time performance of the GPS algorithm may be unsatisfactory. Suppose we want to find the global optimum of a function with several local optima and with a solution set which is a 20-dimensional hyper cubic with a length of 10 in each dimension. The number of solutions in the solution set is approximately 10^{20}. Within 10^4 points, it is understandable that the GPS algorithm cannot find the global optimum. However, when the number of simulated points is larger than 10^4, the computational time for calculating the means and variance at one point in the solution set will be long. Therefore, the speed of the GPS algorithm will be slow. Therefore, in practice we recommend that users apply GPS algorithm in solving problems with a dimensionality less than 20. Finding an efficient way to simplify the calculation of mean and variance is one direction of our future research.

5.5. Stopping Criteria

In this subsection we combine the stopping criteria proposed in §4.3 with the GPS algorithm. Figure 14 shows the objective values and the values of all four stopping criteria with respect to the sample size in a typical run of the algorithm for Problem (14) in the left panel and the inventory problem in the right panel, respectively. From these figures, we find that the performances of the first two criteria are similar, and the performances of the other two are also similar.

For Problem (14), the values of all stopping criteria become stable after 10^4 observations and slowly decrease as the sample size further increases. We can let the algorithm stop after the stopping criteria are lower than some prespecified value. For the inventory problem, however, the values of the four stopping criteria fluctuate significantly even after 2×10^4 observations. Notice that the problem has 2.56×10^{10} feasible solutions and, even with 2×10^4 observations, we only simulate at most 2,000 solutions, which may be too small for any stopping criteria to claim global convergence in a reasonable way. Because judging whether a solution is globally optimal requires information about the entire solution set, stopping criteria for this type of problem tend to be very conservative. Therefore, for high-dimensional problems, our stopping criteria may require significantly more observations to stop than to find a near-optimal solution.

5.6. Impact of Dimensionality

To understand how dimensionality affects the performance of the GPS algorithm, we consider the following problem where the objective function is

$$\min_{\mathbf{x}\in\Theta} g(\mathbf{x}) := \frac{200}{\sqrt{(x_1-5)^2+\cdots+(x_k-5)^2+1}} + \frac{50}{\sqrt{(x_1-7)^2+\cdots+(x_k-7)^2+1}}, \tag{15}$$

Figure 12. (Color online) Sample paths of the GRS, SKO, and GPS algorithms for Problem (14).

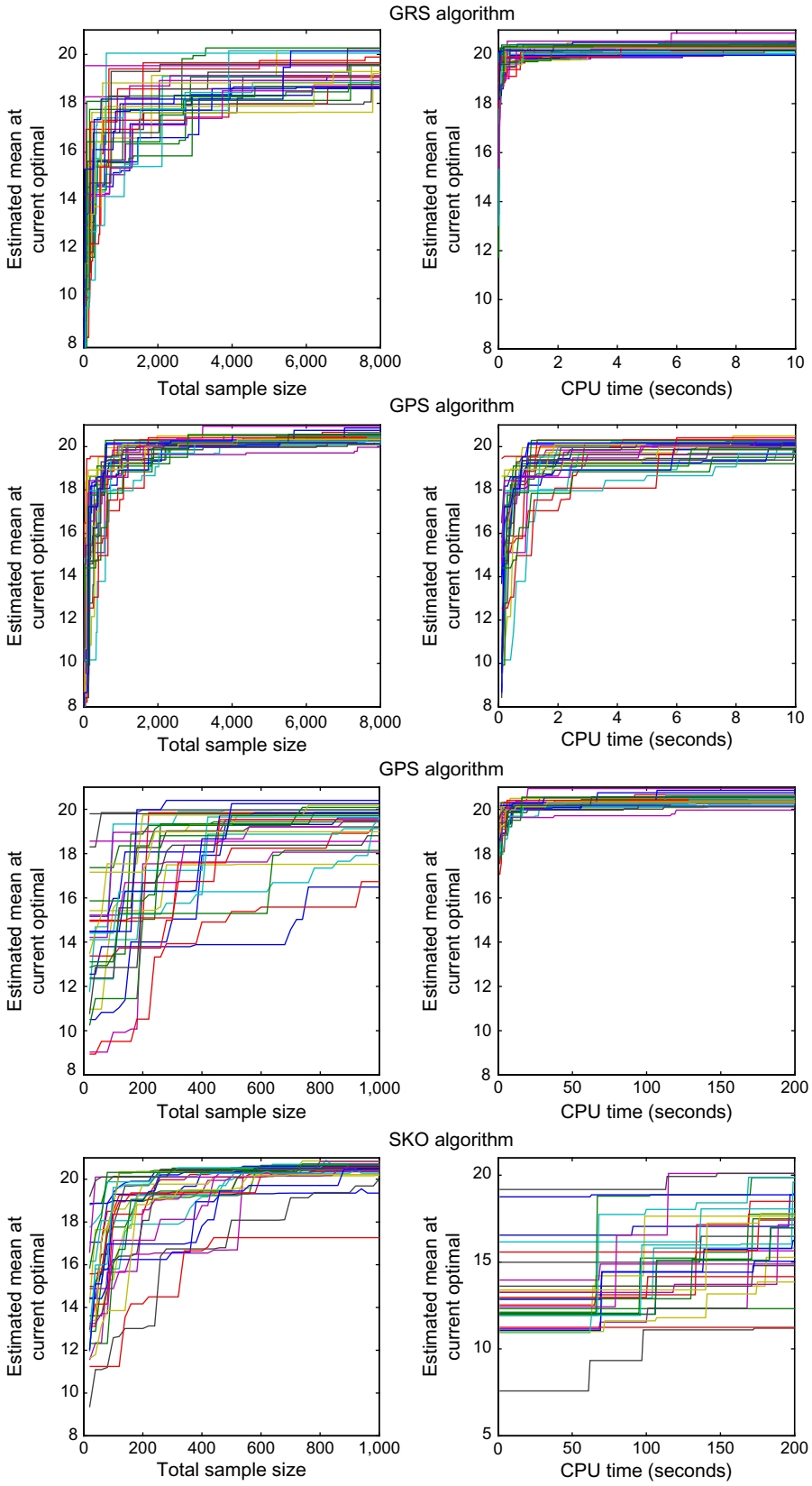


Figure 13. (Color online) Sample paths of the GRS, SKO, and GPS algorithms for the inventory problem.

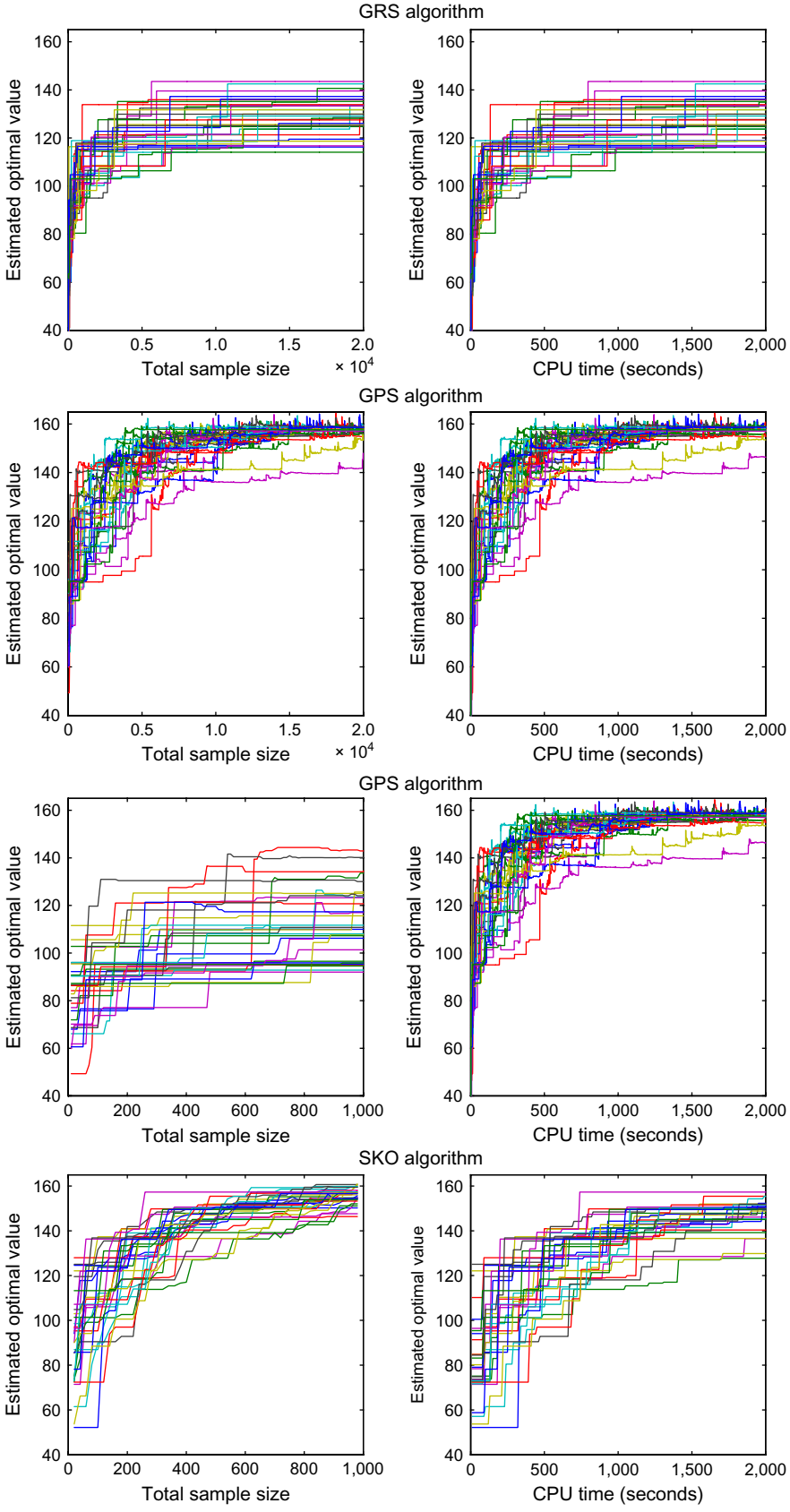
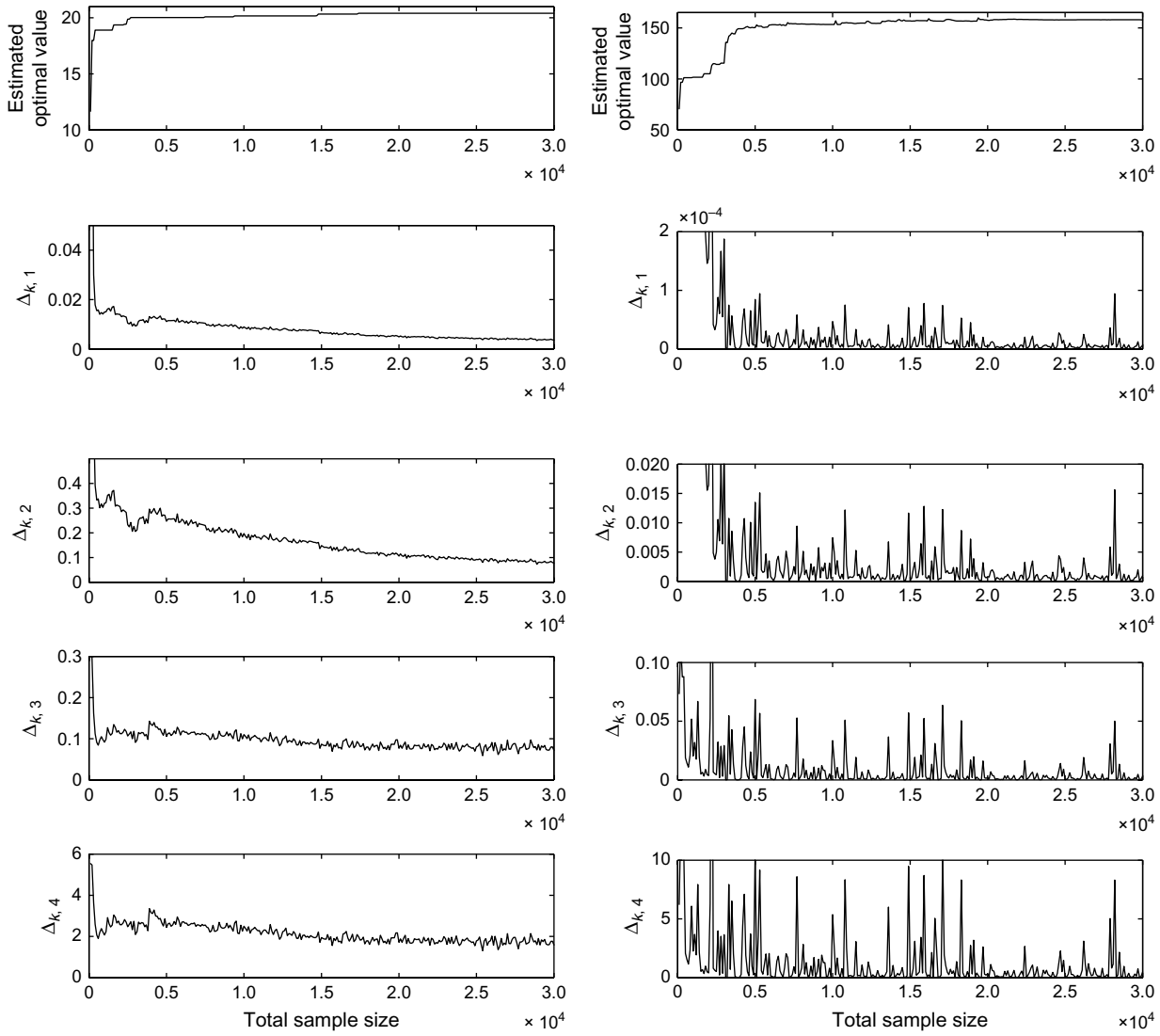


Figure 14. Performance of the GPS algorithm and the stopping criteria for Problem (14) and the inventory problem.



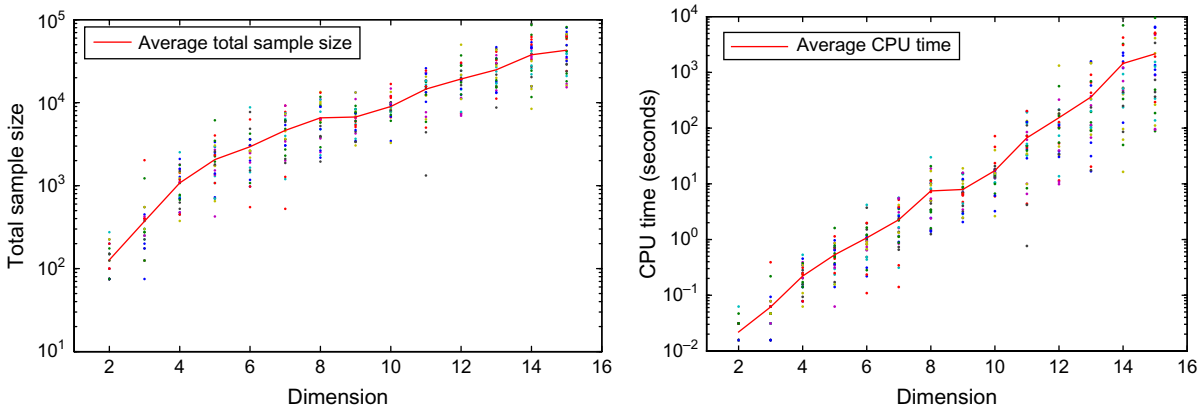
where $\Theta = \{(x_1, \ldots, x_k) \mid x_i = 0.1z_i,\ z_i = 1, \ldots, 100\}$. To make it into a DOvS problem, we assume that the closed-form expression of $g(\mathbf{x})$ is not available and can only be evaluated with noise that is normally distributed with mean 0 and variance 1 for every $\mathbf{x} \in \Theta$. The global optimum of this problem is $(5, \ldots, 5)$, the second largest local optimum is $(7, \ldots, 7)$, and there are also some other local optima.

For the GPS algorithm, we set $\sigma = 5$ with the other settings the same as in Problem (14) and let the algorithm stop when the current sample-best solution locates in $[4.5, 5.5] \times \cdots \times [4.5, 5.5]$. We run the algorithm 30 replications for each dimension from 1 to 15, and plot the total sample sizes and CPU times needed for the algorithm to stop in Figure 15.

Notice that the time to generate an observation from the objective function is negligible. Therefore, the CPU times reported in Figure 15 reflect the computational overhead of the algorithm. From Figure 15 we can see that the computational overhead of the GPS algorithm appears to increase exponentially as the dimension of the problem increases. When the dimension is 10, it takes on average only 20 seconds to solve the problem. But when the dimension is 15, it takes on average over 2,000 seconds (i.e., over 30 minutes). This example suggests that the GPS problem may be suitable for problems with moderate dimensions (e.g., no more than 10–15 dimensions).

The GPS algorithm determines the sampling distribution in each iteration based on a global metamodel of the objective function. Notice that, when the dimension of the problem increases, previously visited solutions tend to be more sparsely allocated in the feasible region, resulting in poorer global metamodels. Therefore, it is not surprising that the performances of the GPS algorithm deteriorate as the dimension of the problem increases. However, we want to emphasize that there are many interesting and important DOvS problems that fall into the range of problems that may be solvable by the GPS algorithm.

Figure 15. (Color online) The total sample sizes and CPU times needed for the algorithm to stop for Problem (15).



6. Conclusions

In this paper we propose a Gaussian process-based approach to constructing sampling distributions that balance the exploitation and exploration trade-off in a seamless way. We then develop the GPS algorithm that implements the sampling distributions, analyze its global convergence, and study its practical performances on two numerical examples. We also propose several stopping criteria that may be used in the GPS algorithm and study their numerical performances.

In the future we plan to extend our work in several directions. First, the sampling distribution can be extended easily to continuous OvS problems. However, the global convergence of the resulted algorithm may be difficult to prove, and we want to study this problem in the future. Second, based on the numerical results, we feel that it may be beneficial to adopt a dynamic estimation scheme in the algorithm to improve the finite time performance. In this paper, we mainly consider the trade-off between the exploitation and exploration and assign a finite number of observations in each iteration to the simulated points. Obviously, estimation is also an important component of stochastic random search algorithms. Third, we believe that the issue of designing stopping criteria for globally convergent random search algorithms is both interesting and important and deserves more study.

Acknowledgments

The authors thank the associate editor and three anonymous referees for their valuable comments and suggestions that significantly improved the paper. A preliminary version of this paper (Sun et al. 2011) was published in the *Proceedings of the* 2011 *Winter Simulation Conference*. This research was partially sponsored by the Hong Kong Research Grants Council [Grant GRF 613011], the Natural Science Foundation of China [Grants 71101111, 71201117, 71371140, 71173153, and 71090404], the Shanghai Pujiang Program, and the Fundamental Research Funds for the Central Universities [2013KJ028].

Appendix A. Proof of Proposition 2

Proof. We first prove the case where $\underline{d}(\mathbf{x}) > 0$. Note that

$$\begin{aligned}\tilde{\sigma}^2(\mathbf{x}) &= \sigma^2(1 - 2\lambda(\mathbf{x})^{\mathrm{T}}\gamma(\mathbf{x}) + \lambda(\mathbf{x})^{\mathrm{T}}\Gamma\lambda(\mathbf{x}))\\ &= \sigma^2\Bigg(1 - 2\sum_{i=1}^{n}\lambda_i(\mathbf{x})h(\|\mathbf{x}-\mathbf{x}_i\|)\\ &\qquad + \sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i(\mathbf{x})\lambda_j(\mathbf{x})h(\|\mathbf{x}_i-\mathbf{x}_j\|)\Bigg)\\ &\geqslant \sigma^2\Bigg(1 - 2\sum_{i=1}^{n}\lambda_i(\mathbf{x})h(\|\mathbf{x}-\mathbf{x}_i\|)\\ &\qquad + \sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i(\mathbf{x})\lambda_j(\mathbf{x})h(\|\mathbf{x}_i-\mathbf{x}\|)h(\|\mathbf{x}-\mathbf{x}_j\|)\Bigg)\\ &\geqslant \sigma^2\Bigg(1 - \sum_{i=1}^{n}\lambda_i(\mathbf{x})h(\|\mathbf{x}-\mathbf{x}_i\|)\Bigg)^2\\ &\geqslant \sigma^2[1 - h(\underline{d}(\mathbf{x}))]^2\end{aligned}$$

where the first inequality follows from Condition 1 and the fact that $\lambda_i(\mathbf{x}) \geqslant 0$ for any i. If $\underline{d}(\mathbf{x}) = 0$, $\mathbf{x} = \mathbf{x}_i$ for some i. Then, $\lambda_i(\mathbf{x}) = 1$ and $\lambda(\mathbf{x}_j) = 0$ for $j \neq i$. Then by some simple algebra, we have $\tilde{\sigma}(\mathbf{x}) = 0$. This completes the proof of the proposition. □

Appendix B. Proof of Proposition 3

Proof. For any $A \subseteq \Theta$,

$$\begin{aligned}\Pr^*\{\mathbf{x} \in A\} &= \Pr^*\{\mathbf{y} \in A \mid \mathrm{U} \leqslant 2\Pr^*\{Y(\mathbf{y}) > c\}\}\\ &= \frac{\Pr^*\{\mathbf{y} \in A, \mathrm{U} \leqslant 2\Pr^*\{Y(\mathbf{y}) > c\}\}}{\Pr^*\{\mathrm{U} \leqslant 2\Pr^*\{Y(\mathbf{y}) > c\}\}}.\end{aligned}$$

Notice that U and $\mathbf{y}$ are independent. Then, we have

$$\begin{aligned}&\Pr^*\{\mathrm{U} \leqslant 2\Pr^*\{Y(\mathbf{y}) > c\}\}\\ &\quad = \sum_{\mathbf{z}\in\Theta}\Pr^*\{\mathrm{U} \leqslant 2\Pr^*\{Y(\mathbf{y}) > c\} \mid \mathbf{y} = \mathbf{z}\}|\Theta|^{-1}\\ &\quad = \sum_{\mathbf{z}\in\Theta}2\Pr^*\{Y(\mathbf{z}) > c\}|\Theta|^{-1} = 1/K.\end{aligned}$$

It follows that

$$\begin{aligned}\Pr^*\{\mathbf{x} \in A\} &= K \cdot \Pr^*\{\mathbf{y} \in A, \mathrm{U} \leqslant 2\Pr^*\{Y(\mathbf{y}) > c\}\}\\ &= K\sum_{\mathbf{y}\in A}2\Pr^*\{Y(\mathbf{y}) > c\}|\Theta|^{-1} = \sum_{\mathbf{y}\in A}f_k(\mathbf{y}),\end{aligned}$$

where the last equality is due to $K = |\Theta| \cdot [\sum_{\mathbf{z}\in\Theta} 2\Pr^*\{Y(\mathbf{y}) > c\}]^{-1}$. Since A is arbitrary, this verifies that the acceptance rejection method is workable. □

Endnotes

1. Similar ideas have also been used in the deterministic black-box optimization literature. For instance, the *P*-algorithm of Calvin and Žilinskas (1999) maximizes the conditional probability of being at least δ better than the current best solution to find the next candidate solution in each iteration, and the EGO algorithm of Jones et al. (1998) maximizes the (conditional) expected improvement over the current best solution to find the next candidate solution in each iteration. In our approach, however, we use the information to build a sampling distribution that guides the random search.

2. As we point out in §2.2, the Gaussian process model is used only to generate an appropriate sampling distribution, and its differentiability has no effect on the convergence of our algorithm.

3. The same problem was also encountered in deterministic black-box optimization algorithms (see, for instance, Jones et al. 1998). However, it is less a problem in those algorithms than in ours because those algorithms often execute a smaller number of iterations and evaluate a much smaller number of solutions.

4. In some cases, for instance, when $G(\mathbf{x})$ is a Bernoulli random variable, it is possible that that the sample variance calculated from a finite sample is zero while the true variance is not. If the zero sample variance is used directly to substitute the true variance in Equation (10), the resulting sampling distribution assigns a zero sampling probability to the solution, thus avoiding the sample mean of the solution being updated, and this may jeopardize the convergence of the algorithm.

5. We implement the SKO algorithm with a *R*-package called DiceOptim to calculate the AEI of each solution and a *R*-package called rgenound to find the solution with the largest AEI value. To compare the three algorithms fairly, we sample 10 solutions using the Latin hypercube sampling from the solution set at the beginning and then sample only one solution in each iteration for all algorithms, because the SKO algorithm only selects the one solution that has the largest AEI value. All other settings of the GRS and GPS algorithms are the same as those used in §§5.1 and 5.2 for Problem (14) and the inventory problem, respectively. Readers may refer to Roustant et al. (2012) for a more detailed description of the DiceOptim package.

References

Alrefaei MH, Andradóttir S (1999) A simulated annealing algorithm with constant temperature for discrete stochastic optimization. *Management Sci.* 45(5):748–764.

Andradóttir S (1995) A method for discrete stochastic optimization. *Management Sci.* 41(12):1946–1961.

Andradóttir S (1996) A global search method for discrete stochastic optimization. *SIAM J. Optim.* 6(2):513–530.

Andradóttir S, Nelson BL (2002) A framework for efficient optimization via simulation. National Science Foundation Grant Proposal, Grant No. DMI-0217690.

Andradóttir S, Prudius AA (2009) Balanced explorative and exploitative search with estimation for simulation optimization. *INFORMS J. Comput.* 21(2):193–208.

Ankenman B, Nelson BL, Staum J (2010) Stochastic kriging for simulation metamodeling. *Oper. Res.* 58(2):371–382.

Baumert S, Ghate A, Kiatsupaibul S, Shen Y, Smith RL, Zabinsky ZB (2009) Discrete hit-and-run for sampling points from arbitrary distributions over subsets of integer hyperrectangles. *Oper. Res.* 57(3): 727–739.

Betrò B (1991) Bayesian methods in global optimization. *J. Global Optim.* 1(1):1–14.

Biles WE, Kleijnen JPC, van Beers WCM, van Nieuwenhuyse I (2007) Kriging metamodeling in constrained simulation optimization: An explorative study. Henderson SG, Biller B, Hsieh M-H, Shortle J, Tew JD, Barton RR, eds. *Proc. 2007 Winter Simulation Conf.* (IEEE, Piscataway, NJ), 355–362.

Calvin J, Žilinskas A (1999) On the convergence of the *P*-Algorithm for one-dimensional global optimization of smooth functions. *J. Optim. Theory Appl.* 102(3):479–495.

Gong W, Ho Y, Zhai W (1999) Stochastic comparison algorithm for discrete optimization with estimation. *SIAM J. Optim.* 10(2):384–404.

Hong LJ, Nelson BL (2006) Discrete optimization via simulation using COMPASS. *Oper. Res.* 54(1):115–129.

Hong LJ, Nelson BL (2007) A framework of locally convergent random search algorithms for discrete optimization via simulation. *ACM Trans. Modeling Comput. Simulation* 17(4):Article 19, 1–22.

Hong LJ, Nelson BL, Xu J (2010) Speeding up COMPASS for high-dimensional discrete optimization via simulation. *Oper. Res. Lett.* 38:550–555.

Hu J, Fu MC, Marcus SI (2008) A model reference adaptive search method for stochastic global optimization. *Comm. Inform. Systems* 8(3):245–276.

Huang D, Allen T, Notz W, Miller R (2006) Sequential kriging optimization using multiple-fidelity evaluations. *Structural and Multidisciplinary Optim.* 32(5):369–382.

Jones DR, Schonlau M, Welch WJ (1998) Efficient global optimization of expensive black-box functions. *J. Global Optim.* 13:455–492.

Kushner HJ (1964) A new method of locating the maximum of an arbitrary multi-peak curve in the presence of noise. *J. Basic Engrg.* 86:97–106.

Law AM, Kelton WD (2000) *Simulation Modeling and Analysis*, 3rd ed. (McGraw-Hill, New York).

Pichitlamken J, Nelson BL (2003) A combined procedure for optimization via simulation. *ACM Trans. Modeling Comput. Simulation* 13(2): 155–179.

Quan N, Yin J, Ng SH, Lee LH (2013) Simulation optimization via kriging: A sequential search using expected improvement with computing budget constraints. *IIE Trans.* 45(7):763–780.

Rasmussen C, Williams C (2006) *Gaussian Processes for Machine Learning* (MIT Press, Cambridge, MA).

Roustant O, Ginsbourger D, Deville Y (2012) Dicekriging, DiceOptim: Two *R* packages for the analysis of computer experiments by kriging-based metamodeling and optimization. *J. Statist. Software* 51(1): 1–55.

Santner TJ, Williams BJ, Notz WI (2003) *The Design and Analysis of Computer Experiments* (Springer, New York).

Sasena M (2002) Flexibility and efficiency enhancements for constrained global design optimization with kriging approximations. Ph.D. thesis, University of Michigan, Ann Arbor.

Shi L, Ólafsson S (2000) Nested partitions method for global optimization. *Oper. Res.* 48(3):390–407.

Stein ML (1999) *Interpolation of Spatial Data: Some Theory for Kriging* (Springer, New York).

Sun L, Hong LJ, Hu Z (2011) Optimization via simulation using Gaussian process-based search. Jain S, Creasey RR, Himmelspach J, White KP, Fu M, eds. *Proc. 2011 Winter Simulation Conf.* (IEEE, Piscataway, NJ), 4134–4145.

Wang H, Pasupathy R, Schmeiser BW (2012) Integer-ordered simulation optimization using R-SPLINE: Retrospective search with piecewise-linear interpolation and neighborhood enumeration. *ACM Trans. Modeling Comput. Simulation* 23(3):Article 17, 1–24.

Xu J, Nelson BL, Hong LJ (2010) Industrial strength COMPASS: A comprehensive algorithm and software for optimization via simulation. *ACM Trans. Modeling Comput. Simulation* 20(1):Article 3, 1–29.

Xu J, Nelson BL, Hong LJ (2013) An adaptive hyperbox algorithm for high-dimensional discrete optimization via simulation problems. *INFORMS J. Comput.* 25(1):133–146.

Yan D, Mukai H (1992) Stochastic discrete optimization. *SIAM J. Control Optim.* 30(3):594–612.

Lihua Sun is an assistant professor in the Department of Economics and Finance at the School of Economics and Management of Tongji University. Her research interests include simulation methodologies, financial engineering, and simulation optimization.

L. Jeff Hong is a Chair Professor of Management Sciences in the College of Business at the City University of Hong Kong. His research interests include stochastic simulation, stochastic optimization, and financial engineering and risk management.

Zhaolin Hu is an associate professor in the Department of Management Science and Engineering at the School of Economics and Management of Tongji University. His current research interests include stochastic optimization, Monte Carlo methods, and risk management.