Proceedings of the 2007 Winter Simulation Conference
S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds.

EXTENSION OF THE DIRECT OPTIMIZATION ALGORITHM FOR NOISY FUNCTIONS

Geng Deng

Department of Mathematics
University of Wisconsin
480 Lincoln Dr.
Madison, WI 53706, U.S.A.

Michael C. Ferris

Computer Sciences Department
University of Wisconsin
1210 W. Dayton Street
Madison, WI 53706, U.S.A.

ABSTRACT

DIRECT (DIviding RECTangles) is a deterministic global optimization algorithm for bound-constrained problems. The algorithm, based on a space-partitioning scheme, performs both global exploration and local exploitation. In this paper, we modify the deterministic DIRECT algorithm to handle noisy function optimization. We adopt a simple approach that replicates multiple function evaluations per point and takes an average to reduce functional uncertainty. A Bayesian sampling scheme is incorporated to determine appropriate numbers of replications.

The noisy version of the DIRECT algorithm is suited for simulation-based optimization problems. The algorithm is a sampling approach, that only uses objective function evaluations. No gradient or Hessian information is required. We have applied the new algorithm to an ambulance base simulation optimization problem.

1 INTRODUCTION

In simulation-based optimization, the objective is often associated with certain performance measure of a simulation system. Several characteristics of such an objective function are relevant here: (a) evaluation has various levels of noise, (b) not differentiable, and (c) computationally expensive to evaluate. In this paper, we consider simulation problems that are subject to simple bounds. The form of the optimization problem is expressed as follows:

$$\min_{x\in\Omega} f(x) = \mathbb{E}[F(x,\xi(\omega))], \tag{1}$$

where

$$\Omega = \{x \in \mathbb{R}^n : l \le x \le u\}.$$

l and u are the lower and upper bounds for the input parameter x, respectively. The underlying function $f(\cdot)$ is unknown and must be estimated. The function $F(\cdot,\xi(\omega))$ is called a sample response function. The output of the function F is affected by a random variable $\xi(\omega)$.

The DIRECT optimization algorithm (Jones, Perttunen, and Stuckman 1993, Jones 2001, Finkel 2003, Finkel 2005) is a global optimization method first motivated by Lipschitz optimization, that has proven to be effective in a wide range of application domains. The algorithm centers around a space-partitioning scheme that divides large hyperrectangles into small ones. The center of each hyperrectangle, considered as a representing point of the hyperrectangle, is evaluated via the objective function. At each iteration, a set of *potentially optimal* hyperrectangles are selected for further divisions. See Figure 1 for an illustration of the algorithm on the Goldstein price function.

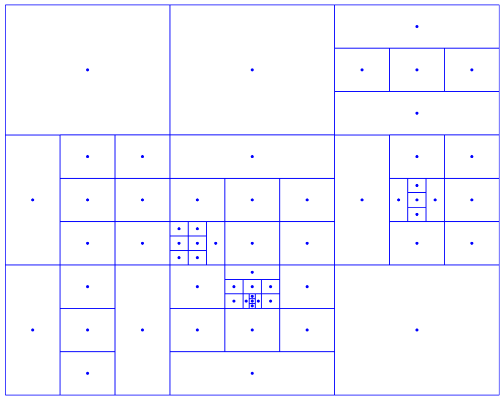

Figure 1: The DIRECT optimization algorithm

When the objective function is subjected to uncertainty, some crucial operational steps of the DIRECT algorithm are affected. For example, the choice of potentially optimal hyperrectangles becomes incorrect because of the noisy function values, possibly misleading the algorithm to search in inferior regions. We modify the original DIRECT algorithm using a simple approach - sampling multiple replications at each point to reduce output uncertainty. We expect the algorithm to proceed correctly as in the deterministic case. However, we must face the issue of handling the tradeoff between two design goals: *efficiency of the algorithm* versus *total computational effort*. Since the objective function is often computationally expensive to evaluate, we must be very cautious in using function evaluations. On the other hand, we need to maintain a certain precision in the functions for correctness of the algorithm. In our modification, we apply Bayesian techniques to derive a posterior distribution for the function output at each point, and incorporate the distribution information into the algorithm to determine an appropriate number of replications to be used.

The remainder of the paper is arranged as follows. In Section 2 we will describe the DIRECT optimization algorithm. In Section 3 we focus on the modifications of the algorithm, including the constructions of a Bayesian posterior distribution and a variance controlling rule to determine the replication numbers. In Section 4, we will present test examples and comparison of the algorithm with other algorithms.

2 THE DIRECT OPTIMIZATION ALGORITHM

The DIRECT algorithm is defined for Lipschitz continuous objective function, a class that includes non-smooth functions. Bounds on the range of the simulation parameter x are required in the design of the algorithm. The feasible region starts as a single hyperrectangle that is internally normalized to a unit hyperrectangle. The algorithm partitions the hyperrectangle into a collection of small hyperrectangles and evaluates the objective function at their center points. Potentially optimal hyperrectangles are identified and passed to the next iteration for further partition and investigation. The DIRECT algorithm will converge to the global optimum of the objective function for dense enough sampling, but the search process may consume a large amount of function evaluations. The typically dimension of a problem, as cited in (Jones, Perttunen, and Stuckman 1993), should be less than 30.

We will give a brief description of the DIRECT algorithm, including two major component steps: (a) partitioning hyperrectangles, and (b) identifying potentially optimal hyperrectangles.

Partitioning hyperrectangles For each hyperrectangle, let $\mathscr{I}$ be the coordinate directions corresponding to the largest side lengths, δ be one third of the largest length, and c be the center point. The function will explore the objective values at the points $c \pm \delta e_i$, for all $e_i \in \mathscr{I}$, where e_i is the ith unit vector. The hyperrectangle will be trisected along the dimensions in $\mathscr{I}$, first along dimensions whose objective values are better. The procedure continues until each point $c \pm \delta e_i$ occupies a single hyperrectangle.

In a two-dimensional case, two possible partitioning scheme are illustrated as follows. Since we only perform trisections, the length of any side in the unit hyperrectangle can possibly be $3^{-k}, k = 1,2,\dots$

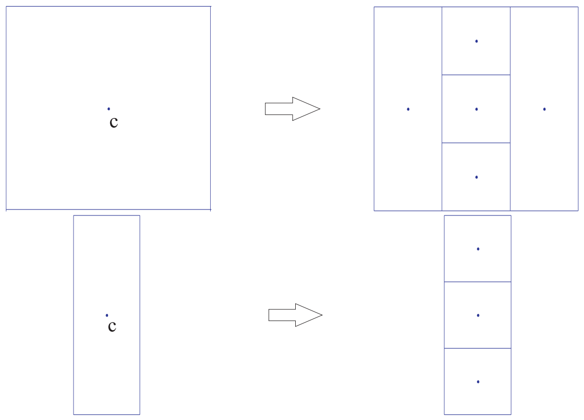

Figure 2: Partitioning hyperrectangles

Identifying potentially optimal hyperrectangles Selection of potentially optimal hyperrectangles combines the purposes of both global and local searches. Let $\mathscr{H}$ be the index set of existing hyperrectangles. For each hyperrectangle $j \in \mathscr{H}$, we evaluate the function value at the center representing point $f(c_j)$ and note the size of the hyperrectangle α_j. The size α_j is computed as the distance from the center point to the corner point. A hyperrectangle $j \in \mathscr{H}$ is said to be potentially optimal ($j \in \mathscr{S}$) if there exists a constant $\widetilde{K}$ such that

$$f(c_j) - \widetilde{K}\alpha_j \leq f(c_i) - \widetilde{K}\alpha_i, \forall i \in \mathscr{H}, \quad (2)$$

$$f(c_j) - \widetilde{K}\alpha_j \leq f_{min} - \varepsilon |f_{min}|. \quad (3)$$

In the above expressions, f_{min} is the lowest function value available and ε is a parameter that balances between global and local search. The parameter is typically nonsensitive and set as 0.0001.

An equivalent process of selecting potential optimal rectangles is illustrated in Figure 3. First sort the hyperrectangles in groups according to the size α. Each hyperrectangle is plotted in the figure as a black dot in accordance with its center function value $f(c_j)$ and size α_j. Criteria (2) and (3) correspond to selecting rectangles on the lower convex hull of the graph (hyperrectangles that are selected are denoted as white dots). The introduction of ε may result in exclusions of good hyperrectangles in the smaller size groups. Thus ε is considered as a balancing parameter between local and global search. As noted from the figure, the best hyperrectangle in the largest size group is always selected. The author claims that the algorithm will eventually converge to the global optimum because the maximum size $\max_j \alpha_j$ decreases to zero and the entire search space is thoroughly explored.

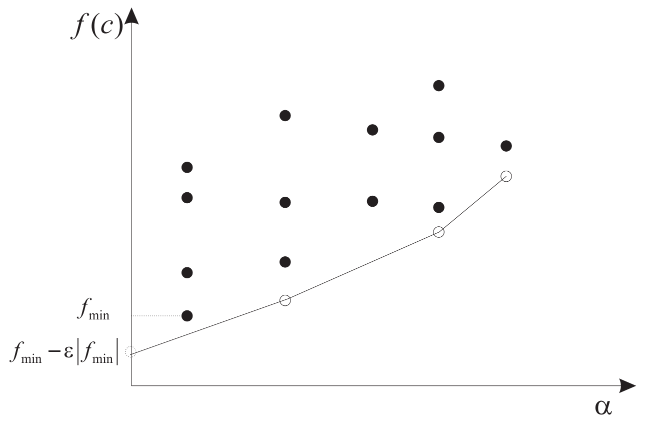

Figure 3: Identifying potentially optimal hyperrectangles

With the two key component steps available, the process of the DIRECT algorithm is straightforward and we summarize the steps of the algorithm below. More details on this algorithm can be found in (Jones, Perttunen, and Stuckman 1993).

The DIRECT optimization algorithm
Given the lower bound l and upper bound u for the optimization problem.

1. Normalize the domain as a unit hyperrectangle.
2. For iterations $k = 1, 2, \ldots$
   (a) Identify the potentially optimal hyperrectangle set $\mathscr{S} \subset \mathscr{H}$.
   (b) Divide the hyperrectangles in $\mathscr{S}$ according to the partitioning scheme.
3. Terminate the algorithm when the total number of function evaluations hits a maximum limit or no improvement of the objective function values is observed after a number of iterations.
4. Return the best point found.

3 MODIFICATIONS

In this section, we will focus on describing our modifications to the deterministic DIRECT algorithm. When noise is present in the objective function output, there are two problematic points in the original DIRECT algorithm.

- **Incorrect potentially optimal hyperrectangle set** $\mathscr{S}$ As we have discussed in Section 2, $\mathscr{S} \subset \mathscr{H}$ is determined according to the function values $f(c_j), j \in \mathscr{H}$ and size α_j (see the equivalent process in Figure 3). When the objective values are volatile, the resulting set $\mathscr{S}$ is unstable. Therefore, the algorithm working with the unstable set will possibly miss important search regions or waste computational effort in redundant regions.
- **Biased solution** The DIRECT algorithm always keeps track of the best point and corresponding best objective value f_{min}. If the objective function is noisy, we cannot assert that the point with the lowest objective value f_{min} is the best point. In such cases, the algorithm may return a solution that is incorrect due to bias from noise.

As we have mentioned above, our primary approach is to sample multiple replications per point and use the averaged value in the algorithm. In order to choose the appropriate number of replications, we first construct a Bayesian posterior distribution for the functional output at each point, and incorporate the distribution information to help determine the appropriate number of replications (See Figure 4). Based on the replication samples, the Bayesian posterior distribution provides not only mean but also variance information of the functional output that can be utilized by the algorithm. At each iteration of the algorithm, our modification focuses on determining and evaluating a necessary number of replications at each point, such that the algorithm can maintain a certain accuracy. The accuracy here explicitly addresses the two problematic issues: the accuracy of the identified promising set $\mathscr{S}$ and the accuracy of the final output of the algorithm.

Based on the mean and variance information, our sampling scheme may generate different numbers of samples for different points. For example, in Figure 4, since Point 1 has a large mean function value (or the volatility is relatively small), 3 replications are enough to obtain the desired accuracy. The number 3 is the minimum number of replications in order to construct the posterior distribution. For Point 3 and Point 5, since their objective values are small (or volatilities are relatively large), more replications are necessary at these points.

3.1 Bayesian Posterior Analysis

There is an extensive literatures on using Bayesian methods in simulation output analysis. Pertinent work includes that of Chick and Inoue (Chick and Inoue 2001, Chick and Inoue 2003), who have implemented Bayesian estimation in ordering discrete simulation systems (see ranking and selection (Chen, Chen, and Yucesan 1999, Kim and Nelson 2003)). Deng and Ferris (Deng and Ferris a, Deng and Ferris b) propose a similar Bayesian analysis to improve a surrogate model based optimization algorithm.

In the Bayesian framework, the unknown mean $\mu(c_j)$ and variance $\sigma^2(c_j)$ of $F(c_j, \xi(\omega))$ are considered as ran-

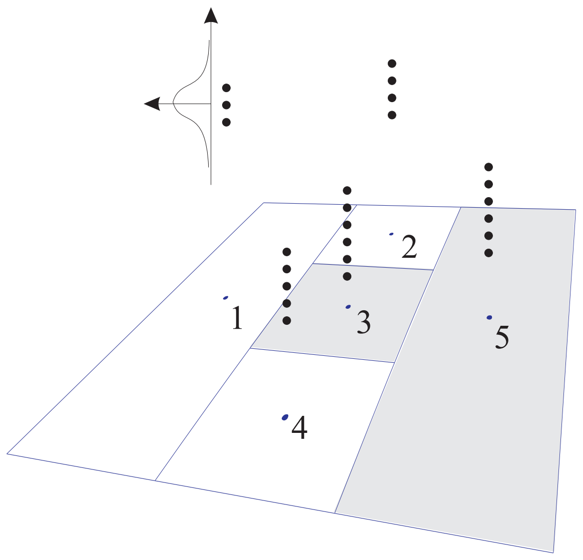

Figure 4: Generate multiple replications at each point and derive Bayesian posterior distributions

dom variables, whose distributions are inferred by Bayes' rule. Since we do not have any prior assumption for the distributions of μ and σ^2, we assign non-informative prior distributions for them. We can estimate the joint posterior distributions of $\mu(c_j)$ and $1/\sigma^2(c_j)$ as

$$\begin{array}{l} \frac{1}{\sigma^2(c_j)}|X \sim Gamma((r_j-1)/2, \hat{\sigma}^2(c_j)(r_j-1)/2), \\ \mu_(c_j)|\sigma^2(c_j), X \sim N(\bar{\mu}(c_j), \sigma^2(c_j)/r_j). \end{array} \tag{4}$$

Here we denote the sample mean and sample variance of the data by $\bar{\mu}(c_j)$ and $\hat{\sigma}^2(c_j)$. The notation ' $|X$' stands for 'posterior distribution with the knowledge of data'. The gamma distribution $Gamma(\alpha,\beta)$ has mean α/β and variance α/β^2. r_j is the number of replications at c_j.

The distribution of the mean value $\mu(c_j)$ is of the most interest to us. When the sample size is large, we can replace the variance $\sigma^2(c_j)$ with the sample variance $\hat{\sigma}^2(c_j)$ in (4), and can asymptotically derive the posterior distribution of $\mu(c_j)|X$ as

$$\mu(c_j)|X \sim N(\bar{\mu}(c_j), \hat{\sigma}^2(c_j)/r_j). \tag{5}$$

Moreover, using an exact computation, the marginal distribution of $\mu(c_j)|X$ inferred by (4) (eliminating $\sigma^2(c_j)$) is,

$$\mu(c_j)|X \sim St(\bar{\mu}(c_j), r_j/\hat{\sigma}^2(c_j), r_j-1), \tag{6}$$

where a random variable with Student's t-distribution $St(\mu,\kappa,\nu)$ has a mean μ, precision κ, and degrees of freedom ν. The $St(\mu,\kappa,\nu)$ turns out to be a transformation of a standard t-distribution with ν degrees of freedom,

$$St(\mu,\kappa,\nu) = \frac{1}{\sqrt{\kappa}} St(0,1,\nu) + \mu.$$

(6) is an exact formulation and matches the result of the frequentists' estimation in constructing confidence intervals for $\mu(c_j)$.

The normal formulation (5) is not as precise. It underestimates the output variance when n is small. But compared to t-distribution (6), it is more convenient to manipulate, and the results of both versions turn out to be very close, as we show below. Therefore, in practice, we suggest using the normal distribution formulation (5).

3.2 Monte Carlo Validation

The goal of our approach is to quantify how the randomness in the objective function affects the identification of the promising hyperrectangle set $\mathscr{S}$. This is done by Monte Carlo validations. Note that the set $\mathscr{S}$ is derived using the observed sample means $\bar{\mu}(c_j)$ in place of $f(c_j)$. As we know, increasing the number of replications r_j at c_j can reduce the uncertainty of the set $\mathscr{S}$, but in our procedure we want to control the total number of function evaluations $\sum_j r_j$. We do this by sampling the posterior distribution and use the samples to determine if further replications are needed.

Suppose we carry out N_t Monte Carlo trial experiments. In the ith experiment $i = 1,2,\ldots,N_t$, we sample objective values $\widetilde{f_i}(c_j)$ from the derived posterior distribution $\mu(c_j)|X, j \in \mathscr{H}$ (c.f., (5)) and plug them into the potentially optimal set identification process (e.g., Figure 3) to generate a trial set $\widetilde{\mathscr{S}_i}$. On completion of all the experiments, we come up with a collection of trial sets $\widetilde{\mathscr{S}_1}, \widetilde{\mathscr{S}_2}, \ldots, \widetilde{\mathscr{S}_{N_t}}$. We then test the difference between the set $\widetilde{\mathscr{S}_i}$ and the original set $\mathscr{S}$. Here, we introduce a volatility controlling rule that requires the trial sets to be close to the set $\mathscr{S}$

$$\frac{1}{N_t}\sum_i^{N_t} \frac{card\left(\widetilde{\mathscr{S}_i} \bigcap \mathscr{S}\right)}{card\left(\mathscr{S}\right)} \geq \beta, \tag{7}$$

where β is a threshold value that is normally set as 90%. The function *card* returns the cardinality of a set. The rule is a Monte Carlo validation criterion that guarantees a sufficiently large overlap between $\widetilde{\mathscr{S}_i}$ and $\mathscr{S}$ ($\widetilde{\mathscr{S}_i}$ and $\mathscr{S}$ are 90% alike on average). Increasing r_j should help reduce the volatility of $\mathscr{S}$, and thus increase the possibility of satisfying the rule (7).

One the other hand, satisfying the above rule necessitates an appropriate number of replication r_j. We adopt a sequential resource allocation procedure. That is, we gradually increase the number r_j until the rule (7) is satisfied.

In doing this, we attempt to minimize the total number of function evaluations. We will selectively increase

$$r_j := \gamma \cdot r_j$$

for $j \in \mathscr{R}$, where γ is an inflation factor and

$$\mathscr{R} = \bigcup_{i}^{N_t} \left(\left(\widetilde{\mathscr{S}_i} / \mathscr{S} \right) \bigcup \left(\mathscr{S} / \widetilde{\mathscr{S}_i} \right) \right). \tag{8}$$

The index set $\mathscr{R}$ is the union of the possible potential optimal rectangles in $\widetilde{\mathscr{S}_i}$ and $\mathscr{S}$, but excluding the intersection of them. That implies we only increase r_j for hyperrectangles that are likely to be potentially optimal. Note that if rule (7) is not satisfied, the set $\mathscr{R}$ is nonempty. The rule (7) will be eventually satisfied because when r_j increase to infinity, Monte Carlo samples $\widetilde{f_i}(c_j)$ are nonvolatile (generated from delta distributions), and the trial sets $\widetilde{\mathscr{S}_i}$ become stable.

Procedure to determine a stable set $\mathscr{S}$

Given an initial sample size r_0, a threshold value β, and a number N_t of Monte Carlo experiments. At each iteration of the DIRECT algorithm, the following procedures should be executed before identifying the potentially optimal set $\mathscr{S}$.

1. Generate r_0 function evaluations for each point c_j for pre-estimations of sample mean and sample variance. Set $r_j \leftarrow r_0, j \in \mathscr{H}$.
2. Carry out N_t Monte Carlo experiments to generate $\widetilde{\mathscr{S}_1}, \widetilde{\mathscr{S}_2}, \ldots, \widetilde{\mathscr{S}_{N_t}}$.
3. While the test rule (7) is not satisfied, increase $r_j \rightarrow \gamma \cdot r_j$, for $j \in \mathscr{R}$. $\mathscr{R}$ is defined in (8). Repeat Step 2.
4. Return the potentially optimal set $\mathscr{S}$.

Previously evaluated function values can certainly be reused.

Our modification does not change the fact that one of the rectangles in the largest size group is divided. Therefore, the division process will finally generate a dense collection of points filling the entire domain. At the return of the algorithm, the point with the lowest averaged sample mean (or averaged sample response function value) is selected. We cannot guarantee the selected point is the real best point (in term of the underlying function $f(x)$) among the explored points, because the sample mean is still subject to noise. In fact, choosing the best point from the vast candidate points is itself a difficult problem (Kim and Nelson 2003). However, compared to the single replication case in the original method, our approach of returning the point with the best sample mean is much more reliable.

4 NUMERICAL EXAMPLES

In this section, several numerical experiments of the noisy DIRECT algorithm are reported, including a particular simulation optimization problem. We also consider comparing the noisy DIRECT against the standard DIRECT and the Snobfit (Stable Noisy Optimization by Branch and Fit) optimization algorithm (Huyer and Neumaier 2006), both of which are well know global optimization methods.

4.1 Numerical functions

For a numerical objective function $f(x, \xi(\omega))$, we consider a special case where $\xi(\omega)$ is an additive 'white noise' term, thus the form of the objective function becomes

$$F(x, \xi(\omega)) = f(x) + \xi(\omega). \tag{9}$$

We assume the noise term is a normally distributed variable $N(0, \sigma^2)$, therefore, the expectation form of the objective is consistent with model (1). Using this formulation, we are able to compare to the known solutions of the deterministic function $f(x)$.

The first test function we employed was the Goldstein price function; see Figure 5 for a contour plot. The function is a two-dimensional function with a global optimum at $(0, -1)$. The global objective value at this point attains 3. We used the bounds $[-2, 2] \times [-2, 2]$ in the optimization methods. The iterates of DIRECT on the noiseless Goldstein price function were shown in Figure 1. We used the following parameter settings for the noisy DIRECT algorithm: the initial sample number $r_0 = 3$, the threshold value $\beta = 90\%$, the number of Monte Carlo trial experiments $N_t = 100$, the inflation factor $\gamma = 1.3$, and the maximum number of replications per point $N_{max} = 100$.

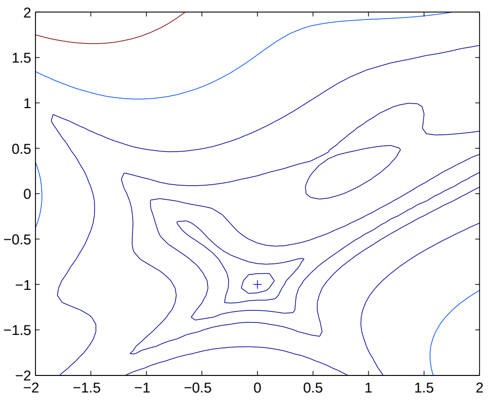

Figure 5: Contour plot of the Goldstein price function

Table 1 shows the performance of the noisy DIRECT algorithm for the case $\sigma^2 = 10$. We terminated the algorithm when the total number of function evaluations reached 3000. The column $\bar{F}(x_k)$ records the best value of the averaged sample response function at iteration k, and $f(x_k)$ records the corresponding best underlying objective function value. In the earlier iterations, the algorithm used relatively few function evaluations. We found only 3 replications were used per point in iterations 1-6. As the iterates got close to the real solution, more and more replications were used to maintain the accuracy of the algorithm. Several points that were near the global solution $(0,-1)$ hit the maximum replication number 100. We generated a replication number plot for the whole optimization process, as shown in Figure 6. It can be shown in this figure that replication numbers increase in promising regions.

Table 1: The performance of noisy DIRECT for the Goldstein price function, with $\sigma^2 = 10$.

Iteration (k)	$f(x_k)$	$\bar{F}(x_k)$	FN
1	200.54	201.03	15
2	200.54	201.03	21
3	14.92	12.68	39
4	14.92	12.68	63
5	3.64	7.16	81
6	3.64	7.16	111
7	5.84	-0.74	298
8	4.55	2.13	380
9	3.55	3.89	1270
10	3.55	3.72	3010

Based on the same Goldstein price example, we compared the noisy version DIRECT algorithm against the deterministic DIRECT and Snobfit. See Table 2 for the comparison results. We present both the normal and t version Bayesian approaches. In order to show the advantage of our automatic replication number selection scheme, we used different fixed replications numbers for other algorithms, for example, 1, 5, 10, 50, and 100. We generated 10 runs of each algorithm and computed the averaged difference in terms of the objective value and the position of solution. (y^* and x^* are the optimal objective value and solution to the Goldstein price function.) Our noisy DIRECT performed the best among all the algorithms. (The difference of using normal and t posterior distributions was small.) As we observed in the table, DIRECT with 1 replication generated worse solutions because the best point is biased by noise, similar to our analysis in Section 3. DIRECT with 50 fixed replications performed very close to our algorithm, and DIRECT with 100 fixed replications performed slightly worse because of early termination of the algorithm (not enough iterations). The automatic scheme indeed saved computation effort in the earlier iterations of the algorithm compared to the fixed replications approaches (c.f., Figure 6). The Snobfit algorithm showed the same issues. The Snobfit algorithm

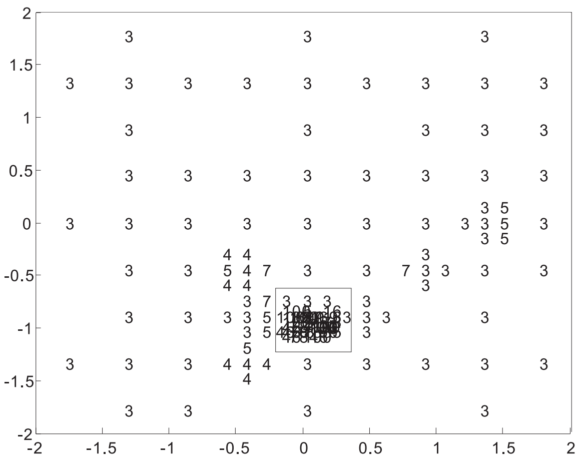
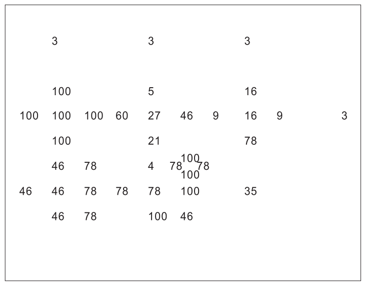

Figure 6: Replication number plot in Goldstein price function optimization. The center region is magnified for a clear view.

in general needed more function evaluations. As we used 3000 maximum function evaluations, Snobfit with 50 and 100 fixed replications cases terminated very early, thus the solutions were significantly worse. We expect the algorithm would perform better given more function evaluations.

We also tested the algorithm on higher dimensional problems. Although the author claims that the suggested dimension of problem should be less than 30, the test problems in (Jones, Perttunen, and Stuckman 1993) have dimensions ranging from 4-6. Here we introduce the 'perm' function, which has an adjustable dimension n

$$f(x) = \sum_{k=1}^{n} \left(\sum_{i=1}^{n} [i^k + \theta][x_i^k - (1/i)^k]\right)^2.$$

This function has a global solution at $x_i = 1/i, i = 1, 2, \ldots, n$, at which the global objective value attains 0. The value θ was set at 0.5, and the range of the region was $[0,1]^n$. In our experiment, we allowed a total of 100,000 function evaluations. Increasing the dimension n made the problem significantly more difficult to solve. In fact, we were not able to obtain a satisfactory solution for 20 dimensional problems. A single run of a 10 dimensional problem (Table 3) showed the same pattern as in the Goldstein price case.

Table 2: Compare noisy DIRECT and other algorithms. (results are based on 10 replications of each algorithm).

	Replication #	Mean $\|f(x_{end}) - f^*\|$	Mean $\|x_{end} - x^*\|$
Noisy DIRECT (normal)	Auto	0.3787	0.0288
Noisy DIRECT (t)	Auto	0.3445	0.0283
DIRECT	1	2.4570	0.0694
	5	1.5045	0.0601
	10	0.6119	0.0432
	50	0.4073	0.0296
	100	0.6474	0.0370
Snobfit	1	2.3231	0.0688
	5	2.0332	0.0802
	10	0.9761	0.0536
	50	11.683	0.1805
	100	44.456	0.5695

Table 3: The performance of noisy DIRECT for the 10-dimensional perm function, with $\sigma^2 = 1$.

Iteration (k)	$f(x_k)$	$\bar{F}(x_k)$	FN
1	372.5	372.6	63
5	45.9	45.3	435
10	5.45	5.84	1299
15	2.3	1.52	2598
20	0.87	0.82	14139
25	0.55	0.28	50131
30	0.30	0.16	107593

4.2 A Simulation Problem

We applied the noisy DIRECT algorithm to solve the ambulance base problem, one of the testbed problems provided in (Pasupathy and Henderson 2006). The problem aims to determine the optimal locations of ambulance bases $p(i), i = 1, 2, \ldots, d$, where d is the number of bases. The emergency calls follow a Poisson arrival process at the rate of λ_a, and all calls are initiated in the region $[0,1]^2$. The location of a call follows a joint triangle distribution function $g(x_1, x_2) = \hat{g}(x_1)\hat{g}(x_2)$, where $\hat{g}(x)$ is a triangle distribution $\hat{g}_{a,b,c}(x)$ (Figure 7).

We assume each ambulance travels at a constant speed v, thus the routing time is twice the distance between the call location and the base divided by the speed v (round trip). The scene time of an ambulance follows an exponential distribution with a location parameter λ_s. The total time of an ambulance at work will be the routing time plus the scene time.

When a call arrives, the nearest free ambulance should respond to the call. If no ambulance is available, the call will be put on wait, and the first ambulance to be freed will respond to this call. The waiting calls are added in FIFO order. In the problem, we aim to find the optimal positions of the ambulance bases such that the total response time is minimized. The response time is consisted of the one-way routing time plus the possible waiting time. We considered two objectives (a) minimize the expected response time, and (b) minimize the maximum response time in a fixed period.

A particular model had the following parameter settings. We considered a simulation period time 500 hours, and $\lambda_a = 0.5$, thus around 1000 calls occurred in each simulation run. $\hat{g}(x)$ used parameters $a = 0, b = 1, c = 0.8$, thus the distribution of calls centered around the location $(0.8, 0.8)$. Vehicle speed was 0.5 and parameter $\lambda_s = 0.1$. The simulation showed that roughly 10% of calls were put on wait. We allowed a total of 20,000 simulation runs, the final positions of the ambulance bases we obtained are plotted in Figure 8. Since the calls were symmetrically distributed about the diagonal of the unit square, the final positions were also approximately symmetric about the diagonal.

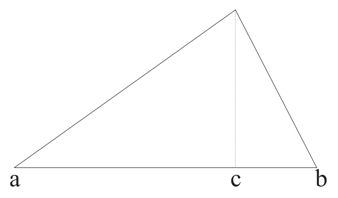

Figure 7: Pdf function of a triangular distribution

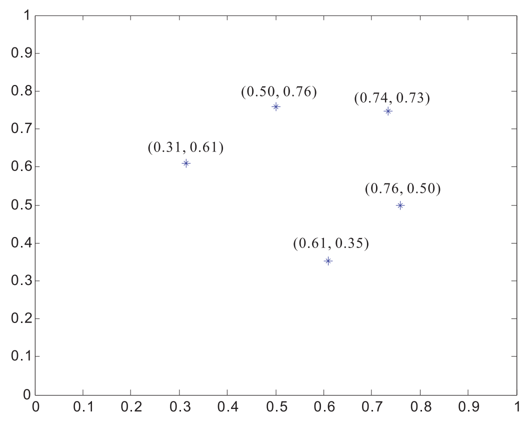

Figure 8: Positions of the ambulance bases

5 CONCLUSIONS

We present a modification to the DIRECT algorithm for noisy function optimization. A Bayesian sampling strategy is introduced to measure the number of replications to use at each point. The appropriate replication number is determined by whether the algorithm can generate an accurate potential optimal hyperrectangle set for further division. The variable number sampling scheme shows advantage over the fixed number sampling approaches. As we have illustrated in numerical examples, the algorithm adaptively has an 'importance sampling' flavor - spending computational effort only in interesting regions, while for sub-interesting regions, the minimum numbers of replications are normally used.

The noisy DIRECT algorithm is suited for global optimization of noisy functions. Since it is a sampling approach, it can handle simulation-based optimization problem whose objective function is a 'black-box' type function.

We are currently developing a generalized two-phase optimization framework for simulation-based optimization. The noisy DIRECT algorithm can be fit into Phase I to serve the purpose of identifying multiple promising regions of interest. Following the return of this algorithm, in Phase II, we apply local search algorithms (for example, the noisy UOBYQA in (Deng and Ferris a)) to refine solutions.

REFERENCES

Chen, H.-C., C.-H. Chen, and E. Yucesan. 1999. An asymptotic allocation for simultaneous simulation experiments. In *Proceedings of the 1999 Winter Simulation Conference*, 359–366.

Chick, S. E., and K. Inoue. 2001. New procedures to select the best simulation system using common random numbers. *Management Science* 47 (8): 1133–1149.

Chick, S. E., and K. Inoue. 2003. New two-stage and sequential procedures for selecting the best simulated system. *Operations Research* 49:1609–1624.

Deng, G., and M. C. Ferris. Adaptation of the UOBYQA algorithm for noisy functions. In *Proceedings of the 2006 Winter Simulation Conference*, ed. L. F. Perrone, F. P. Wieland, B. G. L. J. Liu, D. M. Nicol, and R. M. Fujimoto, 312–319.

Deng, G., and M. C. Ferris. Variable-number sample-path optimization. Submitted to Mathematical Programming.

Finkel, D. E. 2003. DIRECT optimization algorithm user guide. Technical Report CRSC-TR03-11, Center for Research and Scientific Computation, North Carolina State University, Raleigh, NC.

Finkel, D. E. 2005. *Global optimization with the DIRECT algorithm*. Ph. D. thesis, North Carolina State University.

Huyer, W., and A. Neumaier. 2006. Snobfit - stable noisy optimization by branch and fit. submitted.

Jones, D. R. 2001. The DIRECT global optimization algorithm. *Encyclopedia of Optimization*.

Jones, D. R., C. D. Perttunen, and B. E. Stuckman. 1993. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Application* 79 (1): 157–181.

Kim, S.-H., and B. L. Nelson. 2003. Selecting the best system: Theory and methods. In *Proceedings of the 2003 Winter Simulation Conference*, 101–112.

Pasupathy, R., and S. G. Henderson. 2006. A testbed of simulation-optimization problems. In *Proceedings of the 2006 Winter Simulation Conference*, ed. L. F. Perrone, F. P. Wieland, B. G. L. J. Liu, D. M. Nicol, and R. M. Fujimoto, 255–263.

ACKNOWLEDGMENTS

This research was partially supported by Air Force Office of Scientific Research Grant FA9550-04-1-0192, and National Science Foundation Grants DMS-0427689 and IIS-0511905.

AUTHOR BIOGRAPHY

GENG DENG is a PhD student in the optimization group at the University of Wisconsin–Madison. His research is focused on simulation-based optimization, dynamic programming, and applied statistics. He is a member of INFORMS, SIAM and AMS. He can be reached via <geng@cs.wisc.edu>.

MICHAEL C. FERRIS is Professor of Computer Sciences and Industrial Engineering at the University of Wisconsin–Madison. His research interests are in computational algorithms and modeling, coupled with applications in economics, structural and mechanical engineering and medicine. His work emphasizes practical solution of large scale optimization and complementarity problems. He is a member of the Mathematical Programming Society, SIAM and INFORMS. His email and web addresses are <ferris@cs.wisc.edu> and <www.cs.wisc.edu/~ferris>.