Proceedings of the 2017 Winter Simulation Conference
W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, eds.

A COMPUTATIONAL COMPARISON OF SIMULATION OPTIMIZATION METHODS USING SINGLE OBSERVATIONS WITHIN A SHRINKING BALL ON NOISY BLACK-BOX FUNCTIONS WITH MIXED INTEGER AND CONTINUOUS DOMAINS

David D. Linz

Department of Industrial and Systems Engineering
University of Washington
Seattle, WA

Zelda B. Zabinsky

Department of Industrial and Systems Engineering
University of Washington
Seattle, WA

Seksan Kiatsupaibul

Department of Statistics
Chulalongkorn University
Bangkok, THAILAND

Robert L. Smith

Department of Industrial and Operations Engineering
University of Michigan
Ann Arbor, MI

ABSTRACT

We focus on simulation optimization algorithms that are designed to accommodate noisy black-box functions on mixed integer/continuous domains. There are several approaches used to account for noise which include aggregating multiple function replications from sample points and a newer method of aggregating single replications within a "shrinking ball." We examine a range of algorithms, including, simulated annealing, interacting particle, covariance-matrix adaption evolutionary strategy, and particle swarm optimization to compare the effectiveness in generating optimal solutions using averaged function replications versus a shrinking ball approximation. We explore problems in mixed integer/continuous domains. Six test functions are examined with 10 and 20 dimensions, with integer restrictions enforced on 0%, 50%, and 100% of the dimensions, and with noise ranging from 10% to 20% of function output. This study demonstrates the relative effectiveness of using the shrinking ball approach, demonstrating that its use typically enhances solver performance for the tested optimization methods.

1 INTRODUCTION

Due to the increasing application of stochastic simulations in a variety of practical applications, there is increasing interest in simulation optimization in many research communities. Simulation optimization methods typically focus on black-box problems with no known analytical structure (Amaran et al. 2016). Due to this lack of analytical information about the function, optimization is often accomplished through direct observations of function values. Furthermore, for problems with stochastic noise, replications of function values are typically averaged to estimate the fitness of generated points. Since simulations are often computationally expensive, large numbers of runs can quickly make problems computationally intractable.

An alternative method for approximating the fitness of a sampled point for a stochastic black-box function is to examine single observations inside the volume of a hypersphere (Baumert and Smith 2002, Andradóttir and Prudius 2010, Kiatsupaibul et al. 2015). The shrinking ball approach uses function values in a neighborhood of each sample point to approximate the fitness at that point. As the sampling proceeds and the number of values in the neighborhood of the point grows, the shrinking ball reduces the average noise of the estimated black-box function. The shrinking ball approach allows for more of the computational

budget to be used to explore the domain while accounting for noise through aggregation inside of the ball. While some initial computational studies have shown this to be effective for hit-and-run algorithms on a continuous domain (Kiatsupaibul et al. 2015), the method has not been explored broadly for domains that are mixed integer/continuous or for a wider class of optimization algorithms.

A number of papers have measured the relative effectiveness of optimization algorithms for black-box functions on discrete or continuous domains (Neumaier et al. 2005, Floudas and Gounaris 2009, Long-Fei and Le-Yuan 2013), including applications such as (Pintér 2002, Ali et al. 2005, Rios and Sahinidis 2013, Nguyen et al. 2014). However, little attention has been given to the relative impact of alternative methods for estimating function response on the performance of optimization algorithms.

This paper describes the application of the shrinking ball approximation method to four different optimizers: Simulated Annealing Pattern Hit-and-Run (SAPHR), Interacting Particle Algorithm Pattern Hit-and-Run (IPAPHR), Particle Swarm Optimization (PSO), and Covariance-Matrix Adaption Evolution Strategy (CMAES). The methods selected constitute a diverse sampling of approaches to black-box optimization that include heuristic approaches (PSO), random search (SAPHR, IPAPHR), and model-based methods (CMAES) as categorized in (Amaran et al. 2016). Each of the four selected methods are tested with both multiple replication and the shrinking ball approach to estimate the function response.

The test functions are non-convex functions on mixed integer/continuous domains. To compare the selected solvers, we apply each solver to six non-convex functions popular in the benchmark literature (Ali et al. 2005, Rios and Sahinidis 2013). Furthermore, to explore the effect of discretization, we test over a range of dimensions (with both continuous and integer values). We also explore different levels of noise. This benchmarking effort extends earlier efforts to explore the effectiveness of the shrinking ball approach, and provides some insight into performance of popular algorithms on noisy functions in mixed integer/continuous domains.

In general, our results suggest, the shrinking ball will usually outperform multiple replications. The performance improvement is more prevalent for low noise, and low dimensions. The percentage of integer variables has little effect. Moreover, this initial benchmark study points to a benefit of using mixed techniques for controlling noise, particularly using single observations in early stages of optimization to improve the exploration of the domain. Some discussion is offered on the relative performance of different shrinking radius rates and opportunities for extending the research.

2 SIMULATION OPTIMIZATION ALGORITHMS USING SHRINKING BALL ESTIMATION

We consider a black-box function f on a closed, bounded, mixed integer-continuous domain. We are interested in the minimization problem as follows:

$$min_x \; E_\omega[f(x,\omega)] \qquad (1)$$
$$\text{s.t. } x \in S$$

where x is a mixed integer/continuous vector that can be decomposed $x = [x'x'']$ such that $x' \in \mathbb{R}^n$ are the continuous decision variables, and $x'' \in \mathbb{Z}^m$ are the integer decision variables, and S is a domain defined by box constraints. The noise is represented by a random variable ω. We assume there is an optimal solution $x^* \in S$ where $E_\omega[f(x^*)] \leq E_\omega[f(x)] \;\; \forall x \in S$.

2.1 Benchmarked Optimization Algorithms

For purposes of this experiment, we examine two random search strategies (SAPHR and IPAPHR), a heuristic population method (PSO), and a model-based evolutionary search (CMAES). These global optimization algorithms are summarized below within a common framework. The first step (1) is a one-time initialization of parameters, followed by a loop that consists of (2) the sampling of new points based on model specifications, (3) the estimation of some or all of the points objective values based on the points in the previous steps,

and (4)-(6) the update of the sampling model based on the points sampled and estimated in the previous steps.

Simulated Annealing Pattern Hit-and-Run (SAPHR)

The simulated annealing algorithm samples from a domain based on a random step followed by an acceptance rejection criteria extending Markov chain Monte Carlo concepts to global optimization. In this experiment Pattern Hit-and-Run is used to generate new points with simulated annealing (Mete et al. 2011). An acceptance probability with a "cooling schedule" determines how likely it is that the candidate point is accepted as the new point for the next iteration. We implement the algorithm with a constant temperature of $T_k = 500$. The algorithm is summarized as follows:

1. **Initialization:** Set an initial temperature parameter T_0, a random starting point x_0, and set $k = 0$.
2. **Generate New Points:** Generate a new candidate point, x'_k, using Pattern Hit-and-Run.
3. **Estimate Function Response:** Approximate $\hat{f}(x'_k)$ either by a sample average of multiple replications or through the shrinking ball approach.
4. **Acceptance\Rejection:** Calculate an acceptance probability as a function of temperature and the estimated objective function values

$$p_{accept} = e^{((\hat{f}(x_k)-\hat{f}(x'_k))/T_k)}$$

   and update x_{k+1} and $\hat{f}(x_{k+1})$ accordingly.
5. **Update Temperature:** $T_{k+1} = 500$.
6. **Stopping Condition:** If a stopping condition is met, end, otherwise set $k = k+1$ and go to Step 2.

The *Pattern Hit-and-Run Generator*, in Step 2, with box search consists of the following steps:

1. *Step 1:* Generate a continuous point uniformly distributed in the interior of a box $[-c_1, c_1] \times \ldots [-c_n, c_n]$ for a step-size pattern.
2. *Step 2:* Generate a random permutation of n coordinate dimensions.
3. *Step 3:* Uniformly select a point on the sample path as the new candidate point from a forward and backward path of permuted directions extending the current point to the boundary of the domain.

These steps are outlined in (Mete et al. 2011, Mete and Zabinsky 2012). For this experiment, we use a box length parameter as $c_i = 0.5 \cdot H$ where H is the longest side of the domain in question (Mete et al. 2011).

Interacting Particle Algorithm Pattern Hit-and-Run (IPAPHR)

The basic simulated annealing method can be extended to a population-based method by generating multiple candidates, as in the interacting particle algorithm. We use pattern hit-and-run as the generator (Molvalioglu et al. 2009, Mete and Zabinsky 2014). The general form of the algorithm is described as follows:

1. **Initialization:** Set an initial temperature parameter T_0 and a set of points as the current points, $x_{0,l}$ where l indexes the number of L "particles", set $k = 0$.
2. **Generate New Points:** Generate L new candidate points, $x'_{k,l}$, using Pattern Hit-and-Run.
3. **Estimate Function Response:** Approximate $\hat{f}(x'_{k,l})$ for each particle $x_{k',l}$, either by a sample average of multiple replications or through the shrinking ball approach.
4. **Acceptance\Rejection:** Calculate an acceptance probability as a function of temperature and previous sampled points with their estimated objective function values

$$p_{accept,l} = \frac{G(x_{k,l}, x'_{k,l})}{\sum_{l=1}^{L} G(x_{k,l}, x'_{k,l})} \qquad \text{where } G(x_{k,l}, x'_{k,l}) = e^{((\hat{f}(x_{k,l})-\hat{f}(x'_{k,l}))/T_k)}$$

   and update $x_{k+1,l}$ and $\hat{f}(x_{k+1,l)}$ accordingly.

5. **Update Temperature:** $T_{k+1} = 500$.
6. **Stopping Condition:** If stopping condition is met, end, otherwise set $k = k+1$ and go to Step 2.

In the interacting particle algorithm, candidate points' acceptance probability is based on its own fitness relative to the acceptance probability of all of the particles (Mete and Zabinsky 2014). For benchmarking, we implement a population size of 100 and a constant temperature setting of 500 which matches the best performances seen in (Mete and Zabinsky 2014).

Particle-Swarm Optimization (PSO)

The Particle Swarm Optimization method is a common heuristic method based on a population of sample points (Kennedy and Eberhart 1995, Poli et al. 2007). This method is based on simple rules where all sampled points in a population move randomly based on the velocities of nearby points.

1. **Initialization:** Select locations for a set of L starting points $x_{0,l}$, with initial velocity vector $v_{k,l} = 0$, and $k = 0$.
2. **Generate New Points:** Move particles based on velocity to new points such that $x_{k+1,l} = x_{k,l} + v_{k,l}$.
3. **Estimate Function Response:** Approximate $\hat{f}(x_{k+1,l})$ for each point either by a sample average of multiple replications or through the shrinking ball approach
4. **Rank Elite Solutions:** Rank the estimated function evaluations. Determine the best point visited across all particles x^B_{k+1} and within each particle, $x^{b_l}_{k+1,l}$.
5. **Update Velocity:** For each l, determine velocity $v_{k+1,l} = \omega \cdot v_{k,l} + \phi_p \cdot Uniform(0,1) \cdot (x^B_{k+1,l} - x_{k+1,l}) + \phi_g \cdot Uniform(0,1) \cdot (x^{b_l}_{k+1,l} - x_{k+1,l})$.
6. **Stopping Condition:** If stopping condition is met, end, otherwise set $k = k+1$ and go to Step 2.

We use the parameter settings found to be effective in (Langerudi 2014), $\phi_p = 2$, $\phi_g = 2$, and $\omega = 0.7$, in order to control how fast the particles move towards local and global optimal points.

Covariance-Matrix Adaption Evolution Strategy (CMAES)

The Covariance-Matrix Adaption Evolution Strategy is a model-based sampling strategy used in a variety of practical applications that employs an evolving multivariate normal distribution for sampling across the domain (Hansen et al. 2003, Hansen and Kern 2004). Based on an evolutionary update of the covariance matrix and weighted update of the means, the algorithm converges towards the better performing solutions while maintaining some variance in the sampling to explore the domain.

1. **Initialization:** Set covariance update parameters and initial covariance matrix $C = I$ along with selected mean points m_0, σ_0 and weights w, set $k = 0$.
2. **Generate New Points:** Sample L points, $x_{k,l}$, from multivariate normal distribution, $Nor(m_k, \sigma_k \cdot C_k)$, based on mean m_k and covariance matrix update $\sigma_k \cdot C_k$.
3. **Estimate Function Response:** Approximate $\hat{f}(x_{k,l})$ for each point either by a sample average of multiple replications or through the shrinking ball approach.
4. **Update Mean Center:** Update the mean such that points are weighted by their estimated function value.
5. **Update Covariance:** Update the covariance matrix $C_{k+1} = U_{covariance}(C_k, m_k, m_{k+1})$.
6. **Update Variance:** Update the variance $\sigma^2_{k+1} = U_{variance}(\sigma_k, m_k, m_{k+1})$.
7. **Stopping Condition:** If stopping condition is met, end, otherwise set $k = k+1$ and go to Step 2.

The $U_{covariance}$ and $U_{variance}$ functions are specified in (Jastrebski and Arnold 2006). This experiment uses a MATLAB implementation available at (Jastrebski and Arnold 2006, Hansen, N. 2011) with preset parameters.

The CMAES algorithm has an implementation for a different way to handle noise by modeling additional points (Hansen 2010) to create an "implicit" noise handling. For purposes of our benchmark, we implement CMAES with the shrinking ball approach, aggregate multiple replications, and the CMAES "implicit" noise handling.

2.2 Estimating Function Response with Noise

The third step in the global optimization algorithms accounts for the function's noise. One option for estimating function response is a sample average using R replications to account for noise:

$$\hat{f}(x_k) = \frac{\sum_{r=1}^{R} f(x_k, \omega_{k,r})}{R}$$

where $\omega_{k,r}$ is the *rth* sample from the random variable ω. For our experiment, we use this estimation method with $R = 20$ as used for sample average estimation in (Kleywegt et al. 2002, Olafsson 2004).

A second approach to function estimation is to use other function values within a "shrinking ball" of radius r_k. The fitness response of the function with noise is estimated by averaging within the ball, such that:

$$B_{r_k}(x_k) = \{\tilde{x} : ||(\tilde{x} - x_k)|| \leq r_k\} \tag{2}$$

$$\hat{f}(x_k) = \frac{\sum_{\tilde{x} \in B_{r_k}(x_k)} f(\tilde{x}, \tilde{\omega})}{|B_{r_k}(x_k)|}$$

where $|B_{r_k}(x_k)|$ are the number of elements in the ball $B_{r_k}(x_k)$, and $\tilde{\omega}$ is the noise factor associated with the single replication at $\tilde{x}$. Typically, the radius of the ball r_k decreases as the algorithm converges. This allows for exploration early on, with noise correction becoming more prominent as the algorithm converges and as the ball shrinks (Kiatsupaibul et al. 2015). For our experiments we use an initial ball radius of $r_0 = 1$ that decreases to $r_k = 0.5$.

3 PERFORMING OPTIMIZATION TESTS ON SAMPLE FUNCTIONS

We use six non-convex test functions from both (Ali et al. 2005) and (Rios and Sahinidis 2013). These functions include Ackley's Function (between $[-30, 30]$ on each dimension), Griewank Function (between $[-600, 600]$ on each dimension), Rastrigin Function (between $[-5.12, 5.12]$ on each dimension), Rosenbrock's Function (between $[-2, 2]$ on each dimension), and Sinusoidal Function both centered and shifted (between $[0, 180]$ on each dimension).

Table 1: The six test functions with the number of dimensions, and the percentages of integer dimension, and the percent level of noise in the function output.

Test Problem Names	**Dimensions**	**Percent Integer Dimensions**	**Noise**
Ackley's Function	10, 20	0%, 50%, 100%	10%, 20%
Griewank Function	10, 20	0%, 50%, 100%	10%, 20%
Rastrigin Function	10, 20	0%, 50%, 100%	10%, 20%
Rosenbrock Function	10, 20	0%, 50%, 100%	10%, 20%
Sinusoidal Function (centered)	10, 20	0%, 50%, 100%	10%, 20%
Sinusoidal Function (shifted)	10, 20	0%, 50%, 100%	10%, 20%

In order to effectively compare algorithms in a noisy context, we include a random noise factor corresponding to uniform noise of 10% and 20% of the function value respectively. This allows us to explore endogenous

noise in the function by converting a deterministic function $f_0(x)$ into a stochastic function $f(x,\omega)$, such that:

$$f(x,\omega) = f_0(x) + f_0(x) \cdot \omega \cdot \eta \tag{3}$$

where in this case $\omega \sim N(0,1)$ and $\eta \in [0,1]$ is a noise parameter that controls the percentage of noise in the output of the function. This general formulation allows us to control the level of noise in the output. All optimization problems are explored on a set of box constraints in 10 and 20 dimensions, with $0\%, 50\%$ and 100% of those dimensions being integer.

As specified in Table 1, there are six types of test functions with two different dimensions, and three types of integer dimensions, and two levels of noise for a total of $6 \times 3 \times 2 \times 2 = 72$ different trials. Each optimizer is run with both multiple replications and shrinking ball, for 5000 function evaluations (whether used in replications or otherwise). Each trial is repeated 30 times with different initial sample points. The experiment was run in MATLAB 2016a on an Intel Core.

4 DISCUSSION OF RESULTS

We tested nine different optimizers, abbreviated as follows.

1. "SAPHR-sb" - Simulated Annealing with Pattern Hit-and-Run with Shrinking Ball
2. "SAPHR-mr" - Simulated Annealing with Pattern Hit-and-Run with multiple replications
3. "IPAPHR-sb" - Interacting Particle Algorithm Pattern Hit-and-Run with Shrinking Ball
4. "IPAPHR-mr" - Interacting Particles Algorithm Pattern Hit-and-Run with multiple replications
5. "PSO-sb" - Particle Swarm Optimizer with Shrinking Ball
6. "PSO-mr" - Particle Swarm Optimizer with multiple replications
7. "CMAES-sb" - CMAES with Shrinking Ball
8. "CMAES-mr" - CMAES with multiple replications
9. "CMAES-nh" - CMAES with implicit noise handling

Two comparison metrics are used to track the solver progress. For a given best point x_k^* found at a certain number of function evaluations k, we record two objective values:

1. The best estimated function response $\hat{f}(x_k^*)$ averaged over 30 runs
2. The "true" value with no noise of the best point found $f_0(x_k^*)$ averaged over 30 runs.

These two measurements provide insight into the progress accomplished by each of the optimization methods by tracking both their approximated and true value of the solutions found. We expect the first value to consistently decrease versus the number of function evaluations, whereas the second may fluctuate due to noise. Differences between the graphs provide initial insight into the affect of noise on the system.

4.1 Plotting Solver Performance

Figure 1 illustrates the metrics $\hat{f}(x_k^*)$ and $f_0(x_k^*)$ averaged over 30 runs for the six test functions in ten dimensions, all continuous dimensions, with 10% noise. The experiment generated 12 such arrangements of figures but only one is included due to space concerns. In Figure 1, for each test function, there are two graphs. The left graph plots the approximated function values, $(\hat{f}(x_k^*))$, and the right graph plots "true" value, $(f_0(x_k^*))$.

First, throughout the plots in Figure 1 we observe a tendency of methods using shrinking ball approximation to obtain better optimal solutions earlier. This improvement is more pronounced for the approximated function values, $\hat{f}(x_k^*)$, but is still generally present when tracking $f_0(x_k^*)$. The improvement demonstrated by the shrinking ball approach over the multiple replication method is present for all optimizers for some

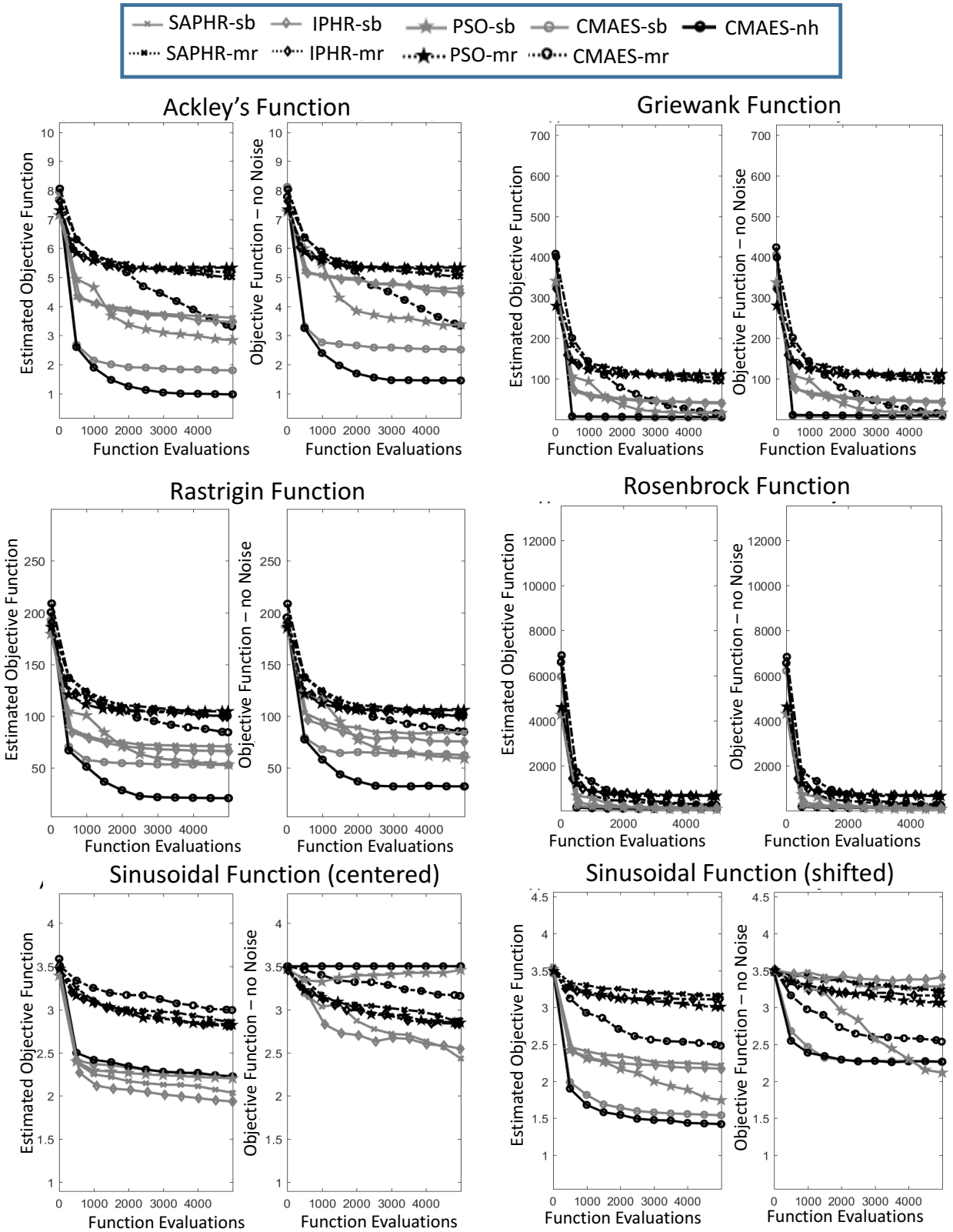

Figure 1: The progress of all nine optimizers for the six test functions in 10 dimensions, zero integer dimensions, and 10% noise. In each pair of plots, the left graph plots the estimated function value, and the right graph plots the true function value for each of the optimizers.

test functions, the Ackley, Griewank, Rastrigin, and Rosenbrock. For the sinusoidal functions (both shifted and centered) only several optimizers using the shrinking ball approach are superior to the multiple replication approach. By observation we note that, with the exception of the centered sinusoidal function, the CMAES optimizer outperforms the other optimizers (for either of the estimation methods). Additionally, the CMAES and PSO optimizers generally demonstrate a greater difference between the performance of optimizers using the shrinking ball approach and optimizers using the multiple replication approach.

We can extend the analysis of the performance of these optimizers to different noise and numbers of dimensions based on the descent of the objective function $f_0(x_k^*)$. We prefer optimizers that provide lower points earlier (at fewer function evaluations). This allows us to determine, by visual examination, which optimizer performed better for each of the 72 different trials averaged over 30 runs. For purposes of tabulation, we measure whether optimizer performance using the shrinking ball crosses below the performance of the multiple replication method before 2500 function evaluations. If the shrinking ball performs better than the multiple replication method by this criteria, we mark (1) otherwise if the multiple replication method outperforms the shrinking ball approach or if the difference is indistinguishable we mark (0). The next section uses this tabulation metric to explore the effect of the shrinking ball with respect to various aspects of the test function.

4.2 Effect of Test Function on Optimizer Performance

The use of the shrinking ball shows a significant effect, across all of the optimizers tested with low noise. We can break down the effectiveness of these optimizers by listing the percent of trials where the shrinking ball approach outperformed that of the multiple replications.

As shown in Table 2, with the exception of the sinusoidal functions, the use of the shrinking ball generally improves performance of the optimizers. From a general examination of each optimizer, the shrinking ball almost always shows a much lower estimated value relative to the multiple replication approach. More importantly, this trend carries over to the true function value $f_0(x^*)$, indicating that the use of a shrinking ball approach allows the algorithms to arrive at a lower objective function value earlier.

Table 2: The percentage over the 12 trials for each test problem where the shrinking ball approach improves the optimizer performance over multiple replications.

	SAPHR-mr	**IPAPHR-mr**	**PSO-mr**	**CMAES-mr**	**CMAES-nh**
Ackley	**50%**	42%	**58%**	**67%**	33%
Griewank	**83%**	**83%**	**100%**	**100%**	25%
Rastrigin	**58%**	**58%**	**83%**	**67%**	8%
Rosenbrock	**100%**	**100%**	**100%**	**100%**	8%
Sinusoidal (shifted)	17%	0%	25%	**58%**	25%
Sinusoidal (centered)	33%	25%	0%	0%	0%

This improved performance is most likely due to the fact that the shrinking ball approach can spend more of its budget on exploring the domain early, where the multiple replication approaches' computational budget is spent on accounting for noise leading to much slower performance of the optimizers tested.

Furthermore, looking at Table 2, we can also note that generally the shrinking ball approach improves performance over PSO-mr and CMAES-mr much more than over the SAPHR-mr or the IPAHR-mr. The CMAES with its own noise-handling, CMAES-nh, outperforms CMAES-sb. Generally both random search approaches (SAPHR and IPAPHR), have very similar performance profiles.

4.3 Effect of Noise on Optimizer Performance

The largest affect on the relative performance of the optimizers is the percentage of noise, η, in the objective function. Here the shift from 10% to 20% noise shows a marked decrease in the effectiveness of solvers that make use of the shrinking ball method to control the noise on the system. As shown in Table 3, the shrinking ball approach improves the performance significantly when there is 10% noise. When there is 20% noise the multiple replication approach is more effective.

Table 3: The percentage of 36 trials for each noise level where the shrinking ball approach improves the optimizer performance.

	SAPHR-mr	**IPAPHR-mr**	**PSO-mr**	**CMAES-mr**	**CMAES-nh**
10% - Noise	**81**%	**72**%	**75**%	**83**%	19%
20% - Noise	33%	31%	47%	47%	14%

The effect of the noise on optimizers increases on higher dimensions, but is moderated on test functions with a higher number of integer dimensions, especially for the SAPHR and IPAPHR optimizers. This generally suggests that the effect of noise might be decreased in the integer domains for solvers that are more conservative in their approach but may be less preferable for solvers which focus on optimal solutions early on.

4.4 Effect of Dimension Number on Optimizer Performance

Largely the increase in domain dimension has a consistent effect across all solvers regardless of the optimizer type, with larger dimensions decreasing the efficiency of the optimizers. Additionally, the solvers show a decreasing difference between the shrinking ball approach and the multiple replication estimation in terms of the performance. In most cases the differences are reduced, however the shrinking ball method still outperforms the multiple replication method on some optimizers as shown in Table 4.

Table 4: The percentage of 36 trials for each number of dimensions where the shrinking ball approach improves the optimizer performance.

	SAPHR-mr	**IPAPHR-mr**	**PSO-mr**	**CMAES-mr**	**CMAES-nh**
10-Dimensional	**67**%	**61**%	**69**%	**69**%	17%
20-Dimensional	47%	42%	**53**%	**61**%	17%

The performance of the algorithms with and without the shrinking ball approach cluster closer together as the number of dimensions increase. Generally at higher dimensions the more ambitious solvers such as the particle swarm with the shrinking ball show greater relative performance than the random search algorithms which show very slow progress inside of the higher dimensions. Nevertheless, there is still a small improvement delivered by using the shrinking ball approach in most trials. However, trials with high noise and high dimension have the multiple replication method outperforming the shrinking ball approach.

4.5 Effect of Number of Integer Dimensions on Optimizer Performance

Another large effect on optimizer performance can be seen by varying the number of integer dimensions in the problems. Generally the number of the integer dimensions improves the overall objective value of the solutions generated by the solvers, causing the optimizers to descend faster towards optimal solutions both in approximation and true value. This is likely due to the more limited domain available to the searches. The introduction of the integer dimensions also causes more separation between the shrinking ball and multiple replication approaches. This is particularly pronounced in the CMAES optimizer.

However, the effect of the number of integer dimensions on the relative effectiveness of the function response method is not large overall, as shown in Table 5. The SAPHR, IPAPHR, and PSO optimizers

Table 5: The percentage of 24 for each number of integer dimensions where the shrinking ball approach improves the optimizer performance.

	SAPHR-mr	**IPAPHR-mr**	**PSO-mr**	**CMAES-mr**	**CMAES-nh**
0%-Integer Dimensions	**54**%	46%	**58**%	**63**%	8%
50%-Integer Dimensions	**54**%	50%	**63**%	**67**%	17%
100%-Integer Dimensions	**63**%	**58**%	**63**%	**67**%	25%

make the largest improvements with the shrinking ball approach in the domains with a higher number of integer dimensions. We speculate that, generally, the pattern hit-and-run sampling techniques allows SAPHR and IPAPHR to more efficiently sample the integer domains which could lead to better use of the algorithm's budget when the shrinking ball method is used.

4.6 Modifying the Shrinking Radius

While the use of the shrinking ball for function response approximation is fairly consistent across the functions with a lower amount of noise and dimension count, most of the improvement demonstrated comes from using single observations to characterize response behavior at a sample point with little influence from other points inside the ball. With the radius set between 1 and 0.5 few sampled points using shrinking ball approximation have previously observed points within the ball $B_{r_k}(x_k)$ for any iteration (with the exception of the CMAES algorithm).

To test an alternative approach, the radius of the shrinking ball was increased to be between 1 and 0.1 of the *longest side* of the box constraints in order to allow for shrinking balls to consistently capture between 10 and 30 other points for all of the methods using the shrinking ball approach. Under this condition however, the shrinking ball approach performed significantly worse than the multiple replication approach across all of the trials. This effect is probably due to a very large radius early in the optimization process which results in a function estimation that may include the entire domain and does not directly distinguish any given point.

5 CONCLUSIONS

The benchmark performed on the selected 72 test functions demonstrates that the shrinking ball is a generally effective method of function estimation in the context of stochastic mixed integer/continuous black-box functions. The initial benchmark demonstrates that for almost all functions with a lower amount of noise the shrinking ball approach has improved the performance of a diverse set of optimizers, demonstrating significant improvements in the solutions developed. Although performance drops off with the increase of dimension and noise, the experiment demonstrates that algorithms which dedicate an increased amount of their early budget to exploration rather than noise control promise a significant improvement to the alternatives.

Increasing the number of dimensions and percentage of noise on the functions significantly inhibits this effect of the shrinking ball approach in improving the performance of the optimizers. While the difference between continuous versus integer domains is less clear, significant performance improvements are observed from the SAPHR and IPAPHR optimizers.

Generally there is a correlation between the improvement seen with the shrinking ball approach and the general effectiveness of an optimizer in solving a given function, with optimizers that make fast descents being improved more by the use of shrinking ball approximation and optimizers that demonstrate more shallow descent showing no improvement or even poorer performance than the multiple replication techniques. This is complemented by the fact that the addition of noise and additional dimensions (which generally detract from solver performance) also detract from the effectiveness of the shrinking ball approach in improving optimizer performance.

The results point to the effectiveness of taking few replications early in the optimization process under most circumstances. However, the issue of accounting for noise seems to be an issue for the shrinking ball approach in higher dimensions. An additional problem arises in high dimensional domains where a small radius fails to capture enough points to sufficiently account for noise and a large radius captures too many points as it searches the domain. A large amount of this effect might be solved by more ambitious cooling schedules for random search optimizers and lower velocity for the particle swarm. This would allow the various optimization methods to focus sampling in smaller regions which would result in more sampling inside balls of smaller radius. Further research directions might focus on matching a decreasing radius and cooling schedule for high noise and high-dimensional functions in order to better understand the trade-off between radius size and the shrinking rate of the ball.

References

Ali, M. M., C. Khompatraporn, and Z. B. Zabinsky. 2005. "A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems". *Journal of Global Optimization* 31 (4): 635–672.

Amaran, S., N. V. Sahinidis, B. Sharda, and S. J. Bury. 2016. "Simulation optimization: a review of algorithms and applications". *Annals of Operations Research* 240 (1): 351–380.

Andradóttir, S., and A. A. Prudius. 2010. "Adaptive random search for continuous simulation optimization". *Naval Research Logistics (NRL)* 57 (6): 583–604.

Baumert, S., and R. Smith. 2002. "Pure random search for noisy objective functions". Technical Report 01-03, University of Michigan, Ann Arbor.

Floudas, C. A., and C. E. Gounaris. 2009. "A review of recent advances in global optimization". *Journal of Global Optimization* 45 (1): 3–38.

Hansen, N. 2010. "Errata / Addenda for A Method for Handling Uncertainty in Evolutionary Optimization With an Application to Feedback Control of Combustion". *IEEE Transactions on Evolutionary Computation* (2).

Hansen, N. 2011. "https://www.lri.fr/~hansen/cmaes_inmatlab.html".

Hansen, N., and S. Kern. 2004. "Evaluating the CMA evolution strategy on multimodal test functions". In *PPSN*, Volume 8, 282–291.

Hansen, N., S. Müller, and P. Koumoutsakos. 2003. "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)". *Evolutionary computation*.

Jastrebski, G., and D. Arnold. 2006. "Improving Evolution Strategies through Active Covariance Matrix Adaptation". In *2006 IEEE International Conference on Evolutionary Computation*, 2814–2821: IEEE.

Kennedy, J., and R. Eberhart. 1995. "Particle swarm optimization". *Neural Networks, 1995. Proceedings., IEEE International Conference on* 4:1942–1948 vol.4.

Kiatsupaibul, S., R. L. Smith, and Z. B. Zabinsky. 2015. "Improving Hit-and-Run with single observations for continuous simulation optimization". In *Winter Simulation Conference (WSC), 2015*, 3569–3576. IEEE.

Kleywegt, A. J., A. Shapiro, and T. Homem-de Mello. 2002. "The Sample Average Approximation Method for Stochastic Discrete Optimization". *SIAM Journal on Optimization* 12 (2): 479–502.

Langerudi, M. F. 2014. "Parameter Selection In Particle Swarm Optimization For Transportation Network Design Problem". *arXiv preprint arXiv:1412.7185*.

Long-Fei, W., and S. Le-Yuan. 2013. "Simulation Optimization: A Review on Theory and Applications". *Acta Automatica Sinica* 39 (11): 1957–1968.

Mete, H. O., Y. Shen, Z. B. Zabinsky, S. Kiatsupaibul, and R. L. Smith. 2011. "Pattern discrete and mixed Hit-and-Run for global optimization". *Journal of Global Optimization* 50 (4): 597–627.

Mete, H. O., and Z. B. Zabinsky. 2012. "Pattern Hit-and-Run for sampling efficiently on polytopes". *Operations Research Letters* 40 (1): 6–11.

Mete, H. O., and Z. B. Zabinsky. 2014. "Multiobjective Interacting Particle Algorithm for Global Optimization". *INFORMS Journal on Computing* 26 (3): 500–513.

Molvalioglu, O., Z. B. Zabinsky, and W. Kohn. 2009. "The interacting-particle algorithm with dynamic heating and cooling". *Journal of Global Optimization* 43 (2-3): 329–356.

Neumaier, A., O. Shcherbina, W. Huyer, and T. Vinkó. 2005. "A comparison of complete global optimization solvers". *Mathematical Programming* 103 (2): 335–356.

Nguyen, A.-t., S. Reiter, and P. Rigo. 2014. "A review on simulation-based optimization methods applied to building performance analysis". *Applied Energy* 113:1043–1058.

Olafsson, S. 2004. "Two-stage nested partitions method for stochastic optimization". *Methodology and Computing in Applied Probability* 6 (1): 5–27.

Pintér, J. D. 2002. "Global Optimization: Software, Test Problems, and Applications". *Handbook of Global optimization* 2:515–569.

Poli, R., J. Kennedy, and T. Blackwell. 2007. "Particle swarm optimization". *Swarm intelligence* 1 (1): 33–57.

Rios, L. M., and N. V. Sahinidis. 2013. "Derivative-free optimization: A review of algorithms and comparison of software implementations". *Journal of Global Optimization* 56 (3): 1247–1293.

AUTHOR BIOGRAPHIES

DAVID D. LINZ is a PhD candidate in the Department of Industrial and Systems Engineering at the University of Washington. His research interests include stochastic optimization, and simulation optimization with applications to healthcare strategy. His e-mail address is ddlinz@uw.edu.

ZELDA B. ZABINSKY is a Professor in the Department of Industrial and Systems Engineering at the University of Washington, with adjunct appointments in the departments of Electrical Engineering, Mechanical Engineering, and Civil and Environmental Engineering. She is an IIE Fellow, and active in INFORMS, serving as General Chair for the INFORMS Annual Meeting held in Seattle in 2007. She has published numerous papers in the areas of global optimization, algorithm complexity, and optimal design of composite structures. Professor Zabinsky's research interests are in global optimization under uncertainty for complex systems. Her email address is zelda@u.washington.edu.

SEKSAN KIATSUPAIBUL is an Associate Professor in the Department of Statistics at Chulalongkorn University, Bangkok, Thailand. His research interests locates at the intersection of operations research and statistics. His e-mail address is seksan@cbs.chula.ac.th.

ROBERT L. SMITH is the Altarum/ERIM Russell D. ONeal Professor Emeritus of Engineering at the University of Michigan, Ann Arbor, Michigan. His research interests lie in the fields of infinite horizon and global optimization. He is an INFORMS Fellow and has served on the editorial boards of Operations Research and Management Science. He was Program Director for Operations Research at the National Science Foundation and Director of the Dynamic Systems Optimization Laboratory at the University of Michigan. His e-mail address is rlsmith@umich.edu.