computations

MDPI

Article

Simulation-Based EDAs for Stochastic Programming Problems

Abdel-Rahman Hedar [1,2,*], Amira A. Allam [3], and Alaa E. Abdel-Hakim [1,4]

1 Department of Computer Science in Jamoum, Umm Al-Qura University, Makkah 25371, Saudi Arabia; adali@uqu.edu.sa
2 Department of Computer Science, Faculty of Computers and Information, Assiut University, Assiut 71526, Egypt; hedar@aun.edu.eg
3 Department of Mathematics, Faculty of Science, Assiut University, Assiut 71516, Egypt; amirahm@science.au.edu.eg
4 Electrical Engineering Department, Assiut University, Assiut 71516, Egypt; alaa.aly@eng.au.edu.eg
* Correspondence: ahahmed@uqu.edu.sa; Tel.: +966-55-0086-411 or +20-10-0070-4940

Received: 15 February 2020; Accepted: 15 March 2020; Published: 18 March 2020

Abstract: With the rapid growth of simulation software packages, generating practical tools for simulation-based optimization has attracted a lot of interest over the last decades. In this paper, a modified method of Estimation of Distribution Algorithms (EDAs) is constructed by a combination with variable-sample techniques to deal with simulation-based optimization problems. Moreover, a new variable-sample technique is introduced to support the search process whenever the sample sizes are small, especially in the beginning of the search process. The proposed method shows efficient results by simulating several numerical experiments.

Keywords: estimation of distribution algorithms; simulation-based optimization; stochastic programming; variable sample path

1. Introduction

Realistic systems often lack a sufficient amount of real data for the purposes of output response evaluation. This fact represents a blocking obstacle to the design process of most of the optimization techniques. However, several processes require optimization at some stage, e.g., engineering design, medical treatment, supply chain management, finance, and manufacturing [1–9]. Therefore, real data alternatives are investigated. For instance, data augmentation is used to expand small-sized sets by applying some transformations to these given real sets [10,11]. Nonetheless, the nature and size limitation of some real data set do not guarantee sufficient diversity of the generated augmented data. Therefore, simulated data are a good choice either for these cases or even other cases. Recently, the combination between simulation and optimization has drawn much attention. The term "*simulation-based optimization*" is commonly used instead of "*stochastic optimization*," since almost all recent simulation software package contain an optimization procedure for creating applicable simulation methods [12,13].

Simulation-based optimization has been used for optimization of real world problems accompanied with some kind of uncertainties in the form of randomness. These problems may need to deal with systems and models, which are highly nonlinear and/or have large-scale dimensions. These kinds of problems can be classified as stochastic programming problems, in which a stochastic probability function is used to estimate the underlying uncertainty. A problem of such kind is defined mathematically as follows [14]:

$$\min_{x \in X} \{f(x) = \mathbb{E}[\mathfrak{F}(x, \omega)]\}, \quad (1)$$

where f is a real-valued function defined on search space $X \subseteq R^n$ with objective variables $x \in X$ and ω is a random variable whose probability density function is $\mathfrak{F}$. The choice of the optimal simulation parameters is critical to performance. However, the optimal selection of these parameters is still a quite challenging task. Because the simulation process is complicated, the objective function is subjected to different noise levels, followed by expensive computational evaluation. The simulation complication is characterized by:

- the calculations and time cost of the objective function,
- the difficulty of computing the exact gradient of the objective function, as well as the high cost and time consuming element of calculating numerical approximations of it, and
- the included noise in objective functions.

To deal with these issues, global search methods should be invoked in order to avoid using classical nonlinear programming methods, which fail to solve these types of multiple local optima problems.

Recently, a great interest has been given to using some artificial tools in optimization. Metaheuristics play a great role in either real-life simulation or invoking intelligent learned procedures [15–24]. Metaheuristics exhibit good degrees of robustness for a wide range of problems. However, these methods suffer from slow convergence in cases of complex problems, which leads to high computational cost. This slow convergence may come from the random structures of exploring the global search space of such methods. On the other side, metaheuristics cannot utilize local information to infer promising search directions. On the contrary, a main characteristic of local search methods is characterized by their fast conversion. Local search methods exploit local information to estimate local mathematical or logical movements in order to determine promising search directions. However, the local search method can easily be stuck at local minima, i.e., it is highly probable to miss global solutions. Therefore, design hybrid methods that combine metaheuristics and local search are highly needed to obtain practical efficient solvers [25–27]. Estimation of Distribution Algorithms (EDAs) are promising metaheuristics—in which exploration for potential solutions in search space depends on building and sampling explicit probabilistic models of promising candidates' solutions [28–31].

There are several optimization and search techniques that have been developed to deal with the considered problem. The variable-sample method is one of these techniques [32]. The main idea of the variable-sample method is to convert a stochastic optimization problem to a deterministic one. This is achieved by estimating the objective function that contains random variables at a certain solution by sampling it with several trails. This type of Monte Carlo simulation can obtain approximate values of the objective function. The accuracy of such simulation depends on the sample-size; therefore, using a large enough size is recommended. The variable-sample method controls the variability of such sample-size in order to maintain the convergence of search methods to optimal solutions under certain conditions. A random search algorithm called Sampling Pure Random Search (SPRS) was previously reported in [32] by invoking the variable-sample method. In SPSR, the objective function with random variables is estimated at a certain input by the average of variable-sized samples at that input. Under certain conditions, the SPSR algorithm can converge to local optimal solutions.

More accurate solutions could be obtained in many cases for stochastic programming problem, e.g., [15,18,33]. In this paper, we present a novel metaheuristic method of Estimation of Distribution Algorithm (EDA). The proposed method is combined with the SPRS method [32], and a modified sampling search method called Min-Max Sampling Search (MMSS) in order to deal with the stochastic programming problem. The main modification in the MMSS method is to use a function transformation instead of the average of the function values, especially when the samples size is not sufficiently large. Actually, the proposed EDA-based search method starts searching the space with relatively small sample sizes in order to achieve faster exploration. Subsequently, the EDA-based method increases

sample sizes while the search process progresses. The proposed modified MMSS method restricts the noisy values of the estimated function in small-size samples at the early stages of the search process. Several experiments with their technical discussion have been done in order to test the performance of the proposed methods.

The rest of the paper is organized as follows. The main structures of the estimation of distribution algorithms and variable-sample methods are highlighted in Sections 2 and 3, respectively. In Section 4, we highlight the main components of the proposed EDA-based methods. In Section 5, we investigate parameter tuning of the proposed methods and explain the experimentation work conducted for evaluation purposes. Detailed results and discussions are given in Section 6. Finally, the paper is concluded in Section 7.

2. Estimation of Distribution Algorithms

The EDAs have been widely studied in the context of global optimization [28–30,34]. The use of the probabilistic models in optimization offers some significant advantages comparing with other types of metahuristics [35–38]. Firstly, the initial population is generated randomly to fill the search space. Then, the best individuals are used to build the probabilistic model. Usually, the best individuals are selected in order to push the search space to the promising regions. New candidate solutions replace the old solutions partly or as a whole, where the individual of highest quality is the target, and the quality of a solution is measured by its associated fitness value. Generally, EDAs execute a repeated process of evaluation, selection, model building, model sampling, and replacement.

The main objective of EDA is to improve the estimation of the solution space probability distribution by exploiting samples generated by the current estimated distribution. All generated samples are subjected to a selection method to pick certain points to be used for probability distribution estimation. Algorithm 1 explains the main steps of standard EDAs.

Algorithm 1 Pseudo code for the standard EDAs

1: $g \leftarrow 0$
2: $D_g \leftarrow$ Generate and evaluate M random individuals (the initial population).
3: **do**
4: $D_g^s \leftarrow$ Select $L \leq M$ individuals from D_g according to a selection method.
5: $P_g(x) = P(x|D_g^s) \leftarrow$ Estimate the joint probability distribution of the selected individuals.
6: $D_{g+1} \leftarrow$ Sample and evaluate M individuals from (the new population) $P_g(x)$.
7: $g \leftarrow g + 1$.
8: **until** a stopping criterion is met.

Many studies have been done in continuous EDAs. Generally, there are two major branches in continuous EDAs. One is widely used that is based on the Gaussian distribution model [28,39] and another major based on the histogram model [40]. The Marginal Distribution Algorithm for continuous domains (UMDAc) is a simple real valued EDA which uses a univariate factorization of normal density in the structure of the probabilistic model. It has been introduced by Larrañaga et al. [41,42]. Some statistical tests were carried out in each generation and for each variable in order to determine the density function that best fits the variables. In this case, the factorization of the joint density function is represented as:

$$f_l(\mathbf{x}; \mathbf{\Theta}^l) = \prod_{i=1}^{n} f_l(\mathbf{x_i}; \mathbf{\Theta_i^l}) \tag{2}$$

where Θ^l is a set of local parameters. The learning process to obtain such joint density function is shown in Algorithm 2.

Algorithm 2 Learning the joint density function

while the stopping criterion is not met **do**
  for $i = 1$ to n **do**
    Set D_{l-1}^{Se,X_i} to be the projection of the selected individuals over the i^{th} variable.
    Select, using a hypothesis test, the density function $f_l(\mathbf{x_i}; \mathbf{\Theta_i^l})$ that best fits D_{l-1}^{Se,X_i}.
    Obtain the maximum likelihood estimates for $\Theta_i^l = (\Theta_i^{l,k_1}, \ldots, \Theta_i^{l,k_i})$.
  end for
  At each generation, the form of the learnt joint density function is: $f_l(x; \Theta^l) = \prod_{i=1}^{n} f_l(x_i; \Theta_i^l)$.
end while

3. Variable Sampling Path

The variable-sample (VS) method [32] is defined as a class of Monte Carlo methods that is used to deal with unconstrained stochastic optimization problems. Sampling Pure Random Search (SPRS) is the random search algorithm introduced by Homem-De-Mello [32] that invokes the VS method. The sample of average approximation with a variable sampling size scheme replaces the objective function in each main iteration of the SPRS algorithm. Specifically, an estimation of the objective function can be computed as:

$$\hat{f}(x) = \frac{\mathfrak{F}(x, \omega_1^k) + \cdots + \mathfrak{F}(x, \omega_{N_k}^k)}{N_k}, \tag{3}$$

where $\omega_1^k, \ldots, \omega_{N_k}^k$ are sample values of the random variable of the objective function. Under certain conditions, the SPRS can converge to a local optimal solution [32]. Algorithm 3 demonstrates the formal algorithm of (SPRS) method.

Algorithm 3 Sampling Pure Random Search (SPRS) Algorithm

Generate a point $x_0 \in X$ at random, set the initial sample size N_0, and set $k := 1$.
Generate a point $y \in X$ at random.
Generate a sample $\omega_1^k, \ldots, \omega_{N_k}^k$.
Compute $\hat{f}(x_k)$, $\hat{f}(y)$ using the following formula: $\hat{f}(x) = \frac{1}{N_k}[\mathfrak{F}(x, \omega_1^k) + \cdots + \mathfrak{F}(x, \omega_{N_k}^k)]$.
If $\hat{f}(y) < \hat{f}(x_k)$, then set $x_{k+1} := y$. Otherwise, set $x_{k+1} := x_k$.
If the stopping criteria are satisfied, stop. Otherwise, update N_k, set $k := k + 1$ and go to Step 2.

4. Estimation of Distribution Algorithms for Simulation-Based Optimization

We propose an EDA-based method to deal with the simulation-based global optimization problems. The proposed method combines the EDA with a modified version of the variable-sample method of Algorithm 3. We suggest a modification to the variable-sample method in order to avoid sample low quality resulted by small sample sizes. This modification depends on a function transformation, as shown in Algorithm 4.

4.1. Function Transformation

The search process is affected with sample sizes. Using large sizes is time consuming, while using small ones yielding low-quality approximation of function values. A good search strategy is to start the search with relatively small-size samples and then increase the sample size while the search is going on. To avoid degradation of the search process when the sample sizes are small in the early stages of the search, a new function transformation is proposed by re-estimating the fitness function if the sample size (N) falls under a certain threshold, i.e., $N < N_{small}$ with a predefined N_{small}. Specifically, the samples of size N at solution x are evaluated and sorted. Then, the $N/2$ highest and $N/2$ lowest objective function values of these samples are averaged and denoted by $\hat{f}_{\max}(x)$ and

$\hat{f}_{\min}(x)$, respectively. Then, a new transformed estimation of the objective function at x is computed using the computed values; $\hat{f}_{\max}(x)$ and $\hat{f}_{\min}(x)$:

$$f(x) \approx \hat{f}_t(x) = \phi(\hat{f}_{\max}(x), \hat{f}_{\min}(x)), \tag{4}$$

with suitable transformation function $\phi(\cdot)$. The choice of a function transformation is preformed empirically to select the function whose best performance among several candidate functions. We found that the following function gives the best performance:

$$\hat{f}_t(x) = \mu(\hat{f}_{\max}(x) - \hat{f}_{\min}(x)), \tag{5}$$

where μ is a weight parameter (with $0 < \mu \leq 1$) that depends on the sample size N at the current iteration. Algorithm 4 uses the above-mentioned function transformation to modify the sampling search method in Algorithm 3.

Algorithm 4 Min-Max Sampling Search (MMSS) Algorithm

procedure
  Generate a point $x_0 \in X$ at random, set the initial sample size N_0, and set $k := 1$.
  while the stopping criteria are not satisfied **do**
    Generate a point $y \in X$ at random.
    Generate a sample $\omega_1^k, \ldots, \omega_{N_k}^k$.
    for $i = 1$ *to* N_k **do**
      Compute $\mathfrak{F}(x, \omega_i^k)$
    end for
    if $N_k \leq N_{small}$ **then**
      compute $\hat{f}(x_k), \hat{f}(y)$ using Equation (3),
    else
      compute $\hat{f}(x_k), \hat{f}(y)$ using Equation (5).
    end if
    if $\hat{f}(y) < \hat{f}(x_k)$ **then**
      then set $x_{k+1} := y$,
    else
      set $x_{k+1} := x_k$.
    end if
    Update N_k, and set $k := k + 1$.
  end while
end procedure

4.2. The Proposed EDA-Based Method

The proposed EDA-based method is a combination of estimation of distribution algorithms with the UMDAc technique [42], in which the stochastic function values can be estimated using sampling techniques. In the simulation-based optimization problem, the objective (fitness) function contains a random variable. Therefore, function values can be approximated using variable sample techniques, which is illustrated in two different ways as in the SPRS and MMSS techniques.

Algorithm 5 illustrates the main steps of the proposed EDA-based method. The initialization process in the EDA has been applied to create M diverse individuals within the search area using a diversification technique similar to those used in [24,43,44]. Besides this exploration process, a local search has been applied to each individual to improve the characteristics of the initial population. The EDA selection process uses the standard EDA selection scheme, in which the best L individuals with the best fitness function values have been chosen from P_g generation to be survive to the next

generation P_{g+1}. In order to improve the performance of the search process, an intensification step has been applied to the best solution in each generation.

Algorithm 5 The proposed EDA-based Algorithm

1: **Initialization.**
   a: Create an initial population D_0 of M individuals.
   b: Evaluate the function values, apply local search to improve each individual.
   c: Set the generation counter $g := 0$.
2: **Main Loop.** Repeat the following steps until the stopping criterion is met
   a: Select the best $L \leq M$ individuals D_g^s.
   b: Estimate joint density function $f(\mathbf{x}; \mathbf{D_g^s})$ of the selected individuals using Algorithm 2.
   c: Sample and evaluate $M - L$ individuals D_g^C from $f(\mathbf{x}; \mathbf{D_g^s})$.
   d: Set $D_{g+1} = D_g^C \cup D_g^s$.
   e: Apply a local search to each individual in D_{g+1}.
   f: Set $k := k + 1$.

The proposed EDA-based method can also be used for deterministic objective function according to the way of evaluating the fitness function in Steps 1:b and 2:c of Algorithm 5, and this yields three versions of the proposed EDA-based method:

- EDA-D: If the objective function has no noise, and its values are directly calculated from the function form.
- EDA-SPRS: If the objective function contains random variables and its values are estimated using the SPRS technique.
- EDA-MMSS: If the objective function contains random variables and its values are estimated using the MMSS technique.

The main difference between difference between EDA-SPRS and EDA-MMSS is in the function evaluation step. The sample path method SPRS depends on sample average approximation in the evaluation function process. However, the search process is not sufficiently appropriate with small sample sizes. This problem affects the quality of the final solution. EDA-MMSS algorithm is modified by applying the same main steps of the EDA-SPRS method, which were explained in the previous subsection, except the function evaluation method. A modified sample path method is defined to evaluate the function values in a small sample size by Equation (5), while the SPRS method is accepted when the number of sample size is sufficiently large. Algorithm 5 is used with Algorithm 4 for the evaluation function to deal with a stochastic optimization problem in order to improve the performance of SPRS method with a small sample size.

5. Numerical Experiments

Experiments are conducted to prove the efficiency of the proposed method and its versions. We use various benchmark data sets for evaluation purposes. In this section, we explain the the details of the experimentation procedures.

5.1. Test Functions

The proposed algorithm is tested using several well-known benchmark functions [45–48] in order to check its ground truth of the method efficiency without the effect of noise. A set of classical test functions are listed in Table 1 that are conducting initial tests for parameter setting and performance analysis of the proposed global optimization method. The characteristics of these test functions are diverse enough to cover many kinds of difficulties that usually arise in global optimization problems. For more professional comparisons, another set of benchmark functions (h_1 - h_{25}) with higher degrees

of difficulty were used. Specifically, the CEC 2005 test set [47,48] is also invoked in comparisons; see Appendix A.

For simulation-based optimization, two different benchmark test sets are used. The first one (SBO—Set A) consists of four well-known classical test functions; Goldstein and Price, Rosenbrock, Griewank, and Pinter functions that are used to compose seven test problems (f_1 - f_7) [49–51]. The details of these seven test functions are shown in Table 2. The mathematical definitions of the test functions are given in Appendix B. The other test set (SBO - Set B) contains 13 test functions (g_1–g_{13}) with Gaussian noise ($\mu = 0, \sigma = 0.2$); see Appendix C for the function definitions of this test set.

Table 1. Classical Global Optimization (CGO) test functions.

No.	f	Function Name	n	No.	f	Function Name	n
1	RC	Branin RCOS	2	6	$P^0_{4,0.5}$	Perm	4
2	GP	Goldstein Price	2	7	T_6	Trid	6
3	SC_2	Schwefel	2	8	G_{20}	Griewank	20
4	Z_2	Zakharov	2	9	DP_{25}	DixonPrice	25
5	$S_{4,7}$	Shekel	4	10	AK_{30}	Ackley	30

Table 2. Classical Simulation-Based Optimization (CSBO) test functions.

Function	Name	n	Stochastic Variable Distribution
f_1	Goldstein & Price	2	$N(0, 10)$
f_2	Rosenbrock	5	$N(0, 10)$
f_3	Griewank	2	$N(0, 10)$
f_4	Pinter	5	$N(0, 10)$
f_5	Modified Griewank	2	$N(0, 10)$
f_6	Griewank	2	$Uniform(-17, 17)$
f_7	Griewank	50	$N(0, 10)$

5.2. Parameter Settings

Table 3 shows the used parameters in the proposed EDAs-based method. An empirical parameter tuning process was followed to find the best values of the parameter set. All parameters are tested to obtain standard settings for the proposed algorithms. The maximum number of function evaluations *maxI*, which is used as a termination criteria, has two different values according to whether the problem is stochastic or deterministic. The EDA population size R and number of selected individuals in each iteration S are considered as measurable effects to build an efficient algorithm. While the main task is to make a balance between the exploration process and avoid the high complexity, we use large R to explore the search space, and then continue with limited selected individuals S to control the complexity of building and sampling the model. The theoretical part of choosing the parameters of the EDAs has been discussed in [29]. The number of sample size N_k, which is used in Algorithm 3 and Algorithm 4, is adapted to be started with N_{small} and then gradually increased to reach N_{max}. The threshold of the small sample size N_{small} and μ parameters used in function transformation in Label 4 are chosen experimentally.

Table 3. Parameter settings.

Parameter	Definition	Best Value
R	Population size	60
S	No. of selected individual size	$0.25 \times R$
$N_{\min}$	The initial value for the sample size	50
$N_{\max}$	The maximum value for the sample size	5000
N_{small}	The threshold of small sample sizes	300
μ	The parameter of the function transformation	$0.5 \times N/N_{\max}$
$maxI$	Max no. of function evaluations for GO	$10{,}000n$
	Max no. of function evaluations for SBO	1,000,000
Runs	No. of independent runs in each experiment	25

5.3. Performance Analysis

In this section, we compare the performance of EDA-SPRA and EDA-MMSS methods. The results are reported in Table 4; these results are the average of errors for obtaining the global optima over 25 independent runs with 500,000 maximum objective function evaluations for each run. In the experimentation, the errors of obtained solutions are denoted by f_{Gap}, which is defined as:

$$f_{Gap} = |(f(x) - f(x^*)|, \tag{6}$$

where x is the best solution obtained by the methods, and x^* is the optimal solution. The results represented in Table 4 show the effect of the function transportation step on the final result. Table 4 reveals that the performance of EDA-MMSS method is better than that of EDA-SPRA at obtaining better solutions in average. Actually, the EDA-MMSS code could reach better average solutions than the EDA-SPRA code in five out of seven test functions. This favors the efficiency of the proposed sampling technique.

Table 4. The f_{Gap} averages and the best solutions obtained by the EDA-SPRA and EDA-MMSS using the SBO Test Set A.

	EDA-SPRS		EDA-MMSS	
f	**Best**	**Average**	**Best**	**Average**
f_1	5.06×10^{-2}	1.64×10^{-1}	6.61×10^{-3}	6.15×10^{-2}
f_2	2.52×10^{0}	3.11×10^{0}	4.50×10^{0}	7.69×10^{0}
f_3	3.75×10^{-3}	4.24×10^{-1}	2.90×10^{-2}	3.22×10^{-1}
f_4	1.39×10^{0}	1.62×10^{1}	9.34×10^{0}	2.23×10^{1}
f_5	3.59×10^{-2}	3.02×10^{-1}	9.11×10^{-4}	2.11×10^{-1}
f_6	9.40×10^{-2}	4.77×10^{-1}	2.91×10^{-3}	2.87×10^{-1}
f_7	3.14×10^{0}	4.56×10^{0}	2.25×10^{0}	2.99×10^{0}

The Wilcoxon rank-sum test [52–54] is used to measure the performance of all the methods compared. This test is known as a non-parametric test [55,56], which our experiments support. The statistical measures used in this test are the sum of rankings obtained in each comparison and the p-value associated. Typically, the differences d_i between the performance values of the two methods on i-th out of n results are calculated. Afterwards, based on the absolute values of these differences, they are ranked. The ranks R^+ and R^- are computed as follows:

$$R^+ = \sum_{d_i>0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i),$$

$$R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i).$$

The statistics of the rank-sum test for an initial comparison between the proposed EDA-SPRS and EDA-MMSS methods are shown in Table 5. Although the EDA-MMSS method could obtain better solutions in four out of seven than the EDA-SPRS method, there is no significant difference between the two compared methods at significance level 0.05.

Table 5. Wilcoxon rank-sum test for the results of Table 4.

Comparison Criteria	Compared Methods	R^+	R^-	p-Value	Best Method
Best Solutions	EDA-SPRS, EDA-MMSS	14	14	0.7104	–
Average Errors	EDA-SPRS, EDA-MMSS	15	13	0.7104	–

Figure 1 shows the performance of EDA-SPRA and EDA-MMSS for one random run, and the figure illustrates the significant difference of the search behavior between the two methods, and how the restriction process of the noisy part of the EDA-MMSS in the beginning of the search process affects the quality of the individual later.

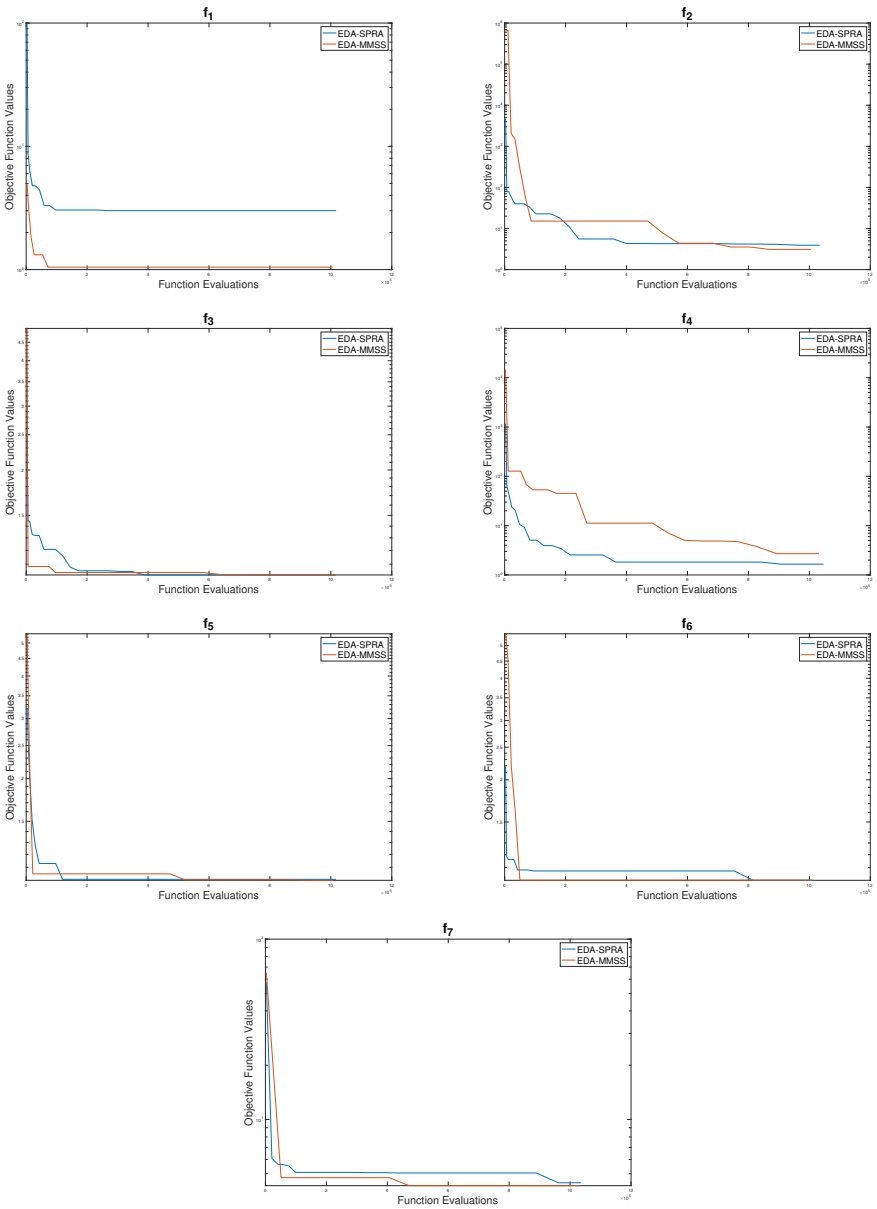

Figure 1. The performance of EDA-SPRA and EDA-MMSS.

The EDA-MMSS method manages to reach better solutions than the EDA-SPRA method in five out of seven cases shown in Figure 1. Therefore, the proposed MMSS technique could help the search method to perform better exploration and reach near global optimal solutions.

6. Results and Discussion

The results of the proposed methods on global optimization and simulation-based optimization problems are discussed in this section.

6.1. Numerical Results on Global Optimization

For global optimization problem, we applied the EDA-D algorithm to ten test problems, which are illustrated in Table 1, before applying the EDA-D algorithm on stochastic optimization problems in order to guarantee its efficiency. Then, it has been compared with other metaheuristic methods. The heuristic solution x is said to be optimal in case of the gap defined in Equation (6) being less than or equal to 0.001. Table 6 reported the average f_{Gap} for 25 independent runs of EDA method for each function, and it is compared with the Directed Scatter Search (DSS) method, which is introduced in [51] with 5000 maximum number of function evaluations for each method. The results shown in Table 6 that the performance of the EDA-D code show promising performance, and its f_{Gap} values show its ability of obtaining global minima for 6 of 10 test problems. The comparison statistics are stated in Table 7, which indicates the similar behavior of the two compared methods at the significance level 0.05.

Table 6. The averages of f_{Gap} obtained by the DSS and EDA-D using the CGO test set.

f	DSS	EDA-D
RC	3.58×10^{-7}	3.59×10^{-7}
$P^0_{4,0.5}$	6.00×10^{-1}	5.02×10^{-4}
GP	4.69×10^{-11}	1.77×10^{-8}
T_6	2.39×10^{-3}	1.00×10^{-3}
SC_2	2.55×10^{-5}	2.50×10^{-3}
G_{20}	5.21×10^{-2}	2.49×10^{1}
Z_2	2.88×10^{-64}	3.48×10^{-11}
DP_{25}	9.43×10^{-1}	1.21×10^{0}
$S_{4,7}$	5.67×10^{-7}	1.00×10^{-4}
AK_{30}	1.04×10^{1}	9.52×10^{0}

Table 7. Wilcoxon rank-sum test for the results of Table 6.

Comparison Criteria	Compared Methods	R^+	R^-	p-Value	Best Method
Average Errors	DSS, EDA-D	22	33	0.7337	–

For a more professional comparison, more sophisticated functions were used to compare the results of the proposed method with those of some benchmark methods. The hard test set test functions h_1–h_{25}, stated in Appendix A, are invoked in testing the performance of the proposed EDA-D method against seven benchmark differential evolutionary methods [57]. The results of the proposed methods and the seven compared methods using the hard test functions h_1-h_{25} with $n = 30$ are reported in Table 8. This table contains average errors with 25 independent trials for each method. Results of the methods used in the comparison are taken from [57]. These comparisons indicate that the proposed method is promising, and its results are similar to those of the methods used in the comparison. Comparative statistics for the results in Table 8 are presented in Table 9 using the rank-sum statistical test. The results of this statistical test indicate that there is no significant difference between the proposed method and the global optimization benchmark methods used in the comparison at the significance level 0.05. This indicates that the proposed method is promising in the field of deterministic

global optimization. This motivates the idea of combining the EDA-D with sampling methods to solve stochastic global optimization problems.

Table 8. Compared average error gabs of global optimization m thods using the hard test set.

h	EDA-D	jDE	SaDE	JADE	CoDE	SSCPDE	CoBiDE	PKDE
h_1	4.63×10^{-12}	0	0	0	6.73×10^{-30}	0	0	0
h_2	4.72×10^{-8}	4.78×10^{-7}	1.12×10^{-5}	8.15×10^{-29}	1.23×10^{-15}	7.79×10^{-14}	1.19×10^{-12}	3.54×10^{-17}
h_3	4.51×10^{-2}	2.03×10^{5}	5.35×10^{5}	8.18×10^{3}	1.20×10^{5}	7.14×10^{4}	7.80×10^{4}	4.89×10^{4}
h_4	5.05×10^{4}	3.44×10^{-2}	1.45×10^{2}	8.39E–16	5.17×10^{-3}	1.59×10^{-3}	8.88×10^{-4}	7.83×10^{-4}
h_5	6.93×10^{3}	4.46×10^{2}	3.13×10^{3}	4.86×10^{-7}	3.64×10^{2}	3.92×10^{2}	3.71×10^{1}	6.29×10^{1}
h_6	7.97×10^{-1}	2.25×10^{1}	4.01×10^{1}	2.58×10^{0}	1.33×10^{-1}	7.80×10^{-9}	1.66×10^{-1}	2.65×10^{-1}
h_7	1.53×10^{-2}	1.16×10^{-2}	1.84×10^{-2}	9.36×10^{-3}	8.86×10^{-3}	4.51×10^{-3}	3.68×10^{-3}	6.07×10^{-3}
h_8	2.00×10^{1}	2.09×10^{1}	2.09×10^{1}	2.09×10^{1}	2.02×10^{1}	2.09×10^{1}	2.07×10^{1}	2.03×10^{1}
h_9	1.52×10^{2}	0	9.95×10^{-2}	0	0	0	0	0
h_{10}	3.18×10^{2}	5.79×10^{1}	4.48×10^{1}	2.43×10^{1}	4.16×10^{1}	2.84×10^{1}	4.34×10^{1}	4.57×10^{1}
h_{11}	3.42×10^{1}	2.83×10^{1}	1.65×10^{1}	2.53×10^{1}	1.26×10^{1}	1.95×10^{1}	5.67×10^{0}	1.36×10^{1}
h_{12}	1.80×10^{2}	1.20×10^{4}	2.17×10^{3}	6.68×10^{3}	3.34×10^{3}	1.64×10^{3}	2.96×10^{3}	3.72×10^{3}
h_{13}	5.76×10^{0}	1.66×10^{0}	3.90×10^{0}	1.48×10^{0}	1.58×10^{0}	2.50×10^{0}	2.64×10^{0}	2.35×10^{0}
h_{14}	1.40×10^{1}	1.30×10^{1}	1.26×10^{1}	1.23×10^{1}	1.24×10^{1}	1.22×10^{1}	1.22×10^{1}	1.23×10^{1}
h_{15}	8.98×10^{2}	3.18×10^{2}	3.74×10^{2}	3.76×10^{2}	4.03×10^{2}	3.30×10^{2}	4.10×10^{2}	3.43×10^{2}
h_{16}	4.67×10^{2}	8.49×10^{1}	7.71×10^{1}	9.63×10^{1}	6.39×10^{1}	4.97×10^{1}	8.42×10^{1}	6.48×10^{1}
h_{17}	4.52×10^{2}	1.39×10^{2}	8.70×10^{1}	1.02×10^{2}	8.50×10^{1}	5.56×10^{1}	6.82×10^{1}	6.83×10^{1}
h_{18}	9.69×10^{2}	9.04×10^{2}	8.78×10^{2}	9.04×10^{2}	9.05×10^{2}	9.00×10^{2}	9.04×10^{2}	9.00×10^{2}
h_{19}	1.02×10^{3}	9.04×10^{2}	8.60×10^{2}	9.04×10^{2}	9.04×10^{2}	9.00×10^{2}	9.04×10^{2}	9.00×10^{2}
h_{20}	9.80×10^{2}	9.04×10^{2}	8.73×10^{2}	9.04×10^{2}	9.04×10^{2}	9.00×10^{2}	9.04×10^{2}	9.00×10^{2}
h_{21}	1.09×10^{3}	5.00×10^{2}	5.43×10^{2}	5.10×10^{2}	5.00×10^{2}	5.00×10^{2}	5.00×10^{2}	5.00×10^{2}
h_{22}	1.24×10^{3}	8.67×10^{2}	9.36×10^{2}	8.64×10^{2}	8.63×10^{2}	8.83×10^{2}	8.54×10^{2}	8.86×10^{2}
h_{23}	1.26×10^{3}	5.34×10^{2}	5.69×10^{2}	5.34×10^{2}	5.34×10^{2}	5.34×10^{2}	5.34×10^{2}	5.34×10^{2}
h_{24}	1.36×10^{3}	2.00×10^{2}	2.00×10^{2}	2.00×10^{2}	2.00×10^{2}	2.00×10^{2}	2.00×10^{2}	2.00×10^{2}
h_{25}	1.37×10^{3}	2.11×10^{2}	2.13×10^{2}	2.11×10^{2}	2.11×10^{2}	2.11×10^{2}	2.10×10^{2}	2.11×10^{2}

Table 9. Wilcoxon rank-sum test for the results of Table 8.

Comparison Criteria	Compared Methods	R^+	R^-	p-Value	Best Method
Average Errors	EDA-D, jDE	263	62	0.2327	–
	EDA-D, SaDE	261	64	0.4151	–
	EDA-D, JADE	269	56	0.1116	–
	EDA-D, CoDE	274	51	0.1683	–
	EDA-D, SSCPDE	273	52	0.1510	–
	EDA-D, CoBiDE	273	52	0.1456	–
	EDA-D, PKDE	275	50	0.1456	–

6.2. Simulation Based Optimization Results

The proposed method has been compared with another metahuristic method to demonstrate its performance in terms of simulation optimization. The EDA-MMSS method has been compared with Evolution Strategies and Scatter Search for a simulation-based global optimization problem, which is introduced in [51]. Table 10 shows the comparison among the proposed method, Directed Evolution Strategies for Simulation-based (DESSP), and Scatter Search for Simulation-Based Optimization (DSSSP). The averages of the function values for the tested functions, and the best values over 25 independent runs for EDA-MMSS, DESSP, and DSSSP are simulated to seven test function problems presented in Table 2 with 500,000 maximum function evaluations, and their processing times are shown in Table 11. From Table 10, the EDA-MMSS method has shown superior performance for this function, especially for high dimensional function f_7.

Table 10. Best and average function values using the SBO Test Set A.

f	EDA-SPRS		EDA-MMSS		DESSP		DSSSP	
	Best	**Average**	**Best**	**Average**	**Best**	**Average**	**Best**	**Average**
f_1	5.06×10^{-2}	1.64×10^{-1}	6.61×10^{-3}	6.15×10^{-2}	5.00×10^{-4}	2.33×10^{-1}	1.46×10^{-2}	2.94×10^{-1}
f_2	2.52×10^{0}	3.11×10^{0}	4.50×10^{0}	7.69×10^{0}	8.05×10^{-1}	3.55×10^{0}	4.08×10^{-1}	6.56×10^{0}
f_3	3.75×10^{-3}	4.24×10^{-1}	2.90×10^{-2}	3.22×10^{-1}	1.00×10^{-3}	3.31×10^{-1}	5.90×10^{-1}	1.04×10^{0}
f_4	1.39×10^{0}	1.62×10^{1}	9.34×10^{0}	2.23×10^{1}	1.44×10^{-1}	3.75×10^{0}	2.75×10^{0}	6.71×10^{0}
f_5	3.59×10^{-2}	3.02×10^{-1}	9.11×10^{-4}	2.11×10^{-1}	4.00×10^{-4}	3.87×10^{-1}	2.82×10^{-1}	1.91×10^{0}
f_6	9.40×10^{-2}	4.77×10^{-1}	2.91×10^{-3}	2.87×10^{-1}	1.00×10^{-5}	2.13×10^{-2}	3.00×10^{-4}	9.21×10^{-2}
f_7	3.14×10^{0}	4.56×10^{0}	2.25×10^{0}	2.99×10^{0}	2.79×10^{1}	3.96×10^{1}	8.41×10^{0}	1.24×10^{1}

Table 11. Averages of processing time (in seconds) for obtaining results in Table 10.

f	f_1	f_2	f_3	f_4	f_5	f_6	f_7
EDA-SPRS	79	10	39	75	10	5	241
EDA-MMSS	85	11	40	76	11	5	251
DESSP	162	20	119	177	25	10	476
DSSSP	414	103	317	307	114	35	683

Table 12 provides a statistical comparisons for the results in Tables 10 and 11. Although there were no significant differences between the results of the proposed method and those of the methods used in the comparison in terms of solution qualities, it is clear that the proposed method obtained better results on solution averages. This indicates the robustness of the proposed method. Moreover, the proposed method shows superior performance in saving processing times as shown in Tables 11 and 12.

Table 12. Wilcoxon rank-sum test for the results of Tables 10 and 11.

Comparison Criteria	Compared Methods	R^+	R^-	p-Value	Best Method
Best Solutions	EDA-SPRS, EDA-MMSS	14	14	0.7104	–
	EDA-SPRS, DESSP	21	7	0.2593	–
	EDA-SPRS, DSSSP	9	19	0.9015	–
	EDA-MMSS, DESSP	16	12	0.3829	–
	EDA-MMSS, DSSSP	11	17	0.8048	–
Average Errors	EDA-SPRS, EDA-MMSS	15	13	0.7104	–
	EDA-SPRS, DESSP	14	14	0.8048	–
	EDA-SPRS, DSSSP	9	19	0.8048	–
	EDA-MMSS, DESSP	22	6	0.2086	–
	EDA-MMSS, DSSSP	18	10	0.8048	–
Processing Time	EDA-SPRS, EDA-MMSS	0.5	27.5	0.6474	–
	EDA-SPRS, DESSP	0	28	0.3141	–
	EDA-SPRS, DSSSP	0	28	0.0169	EDA-SPRS
	EDA-MMSS, DESSP	0	28	0.3642	–
	EDA-MMSS, DSSSP	0	28	0.0169	EDA-MMSS

The last experiment was conducted to compare the results of the proposed methods with those of recent benchmark methods in simulation-based optimization. Eight methods [58] are used in this final comparison and their results are reported in Table 13 using the SBO Test Set B. The proposed methods could obtain better results than seven out of eight methods used in this comparison. Comparative statistics for these results are reported in Table 14 using the rank-sum statistical test. These statistics indicate that there is no difference between the results of the proposed EDA-SPRS and EDA-MMSS methods. Moreover, the proposed methods have overcome 7 out of 8 methods used in the comparison, which is clear from the p-values.

Table 13. Compared average error gabs of simulation-based optimization methods using the SBO Test Set B with $n = 30$.

g	**EDA-SPRS**	**EDA-MMSS**	**DE/rand/1**	**jDE**	**GADS**
g_1	2.73×10^{-1}	5.60×10^{-1}	3.67×10^{0}	4.59×10^{-1}	1.95×10^{0}
g_2	1.79×10^{0}	8.54×10^{0}	5.89×10^{1}	4.25×10^{1}	3.24×10^{1}
g_3	3.67×10^{0}	9.25×10^{0}	6.56×10^{3}	4.27×10^{3}	7.44×10^{3}
g_4	1.63×10^{-1}	4.14×10^{0}	1.02×10^{2}	5.82×10^{1}	6.49×10^{1}
g_5	3.88×10^{-1}	2.82×10^{-1}	9.98×10^{-3}	7.67×10^{-1}	2.76×10^{-2}
g_6	2.00×10^{0}	2.42×10^{0}	4.18×10^{2}	1.99×10^{2}	4.75×10^{2}
g_7	2.40×10^{1}	2.28×10^{1}	6.34×10^{0}	2.08×10^{1}	1.29×10^{1}
g_8	5.67×10^{0}	3.73×10^{1}	1.65×10^{4}	9.19×10^{3}	1.05×10^{4}
g_9	1.07×10^{1}	1.12×10^{1}	5.74×10^{0}	4.65×10^{0}	3.81×10^{0}
g_{10}	2.50×10^{1}	4.26×10^{1}	3.91×10^{2}	2.90×10^{2}	2.31×10^{2}
g_{11}	3.41×10^{1}	4.37×10^{1}	6.36×10^{3}	4.12×10^{3}	7.50×10^{3}
g_{12}	8.23×10^{3}	4.51×10^{3}	7.89×10^{3}	8.22×10^{3}	6.09×10^{3}
g_{13}	9.64×10^{-1}	6.19×10^{-1}	1.18×10^{0}	9.91×10^{-1}	1.46×10^{0}
g	DERSFTS	OBDE	NADE	MUDE	MDE-DS
g_1	1.10×10^{0}	3.35×10^{0}	2.99×10^{-1}	2.59×10^{-1}	0
g_2	5.67×10^{1}	5.69×10^{1}	3.29×10^{1}	2.69×10^{1}	8.53×10^{-3}
g_3	6.84×10^{3}	8.23×10^{3}	3.32×10^{3}	2.36×10^{3}	7.08×10^{-4}
g_4	1.06×10^{2}	1.45×10^{2}	4.53×10^{1}	3.68×10^{1}	7.92×10^{-2}
g_5	7.84×10^{-1}	4.16×10^{-2}	8.30×10^{-1}	7.69×10^{-1}	5.41×10^{-4}
g_6	3.95×10^{2}	5.21×10^{2}	1.65×10^{2}	1.46×10^{2}	1.41×10^{-3}
g_7	8.52×10^{0}	9.27×10^{0}	2.46×10^{1}	2.32×10^{1}	2.02×10^{0}
g_8	1.66×10^{4}	2.15×10^{4}	6.84×10^{3}	5.28×10^{3}	7.81×10^{-1}
g_9	3.92×10^{0}	4.25×10^{0}	5.05×10^{0}	5.41×10^{0}	1.39×10^{0}
g_{10}	3.83×10^{2}	3.78×10^{2}	1.96×10^{2}	2.00×10^{2}	1.89×10^{-2}
g_{11}	6.12×10^{3}	7.26×10^{3}	3.76×10^{3}	2.49×10^{3}	2.60×10^{1}
g_{12}	1.01×10^{4}	8.03×10^{3}	5.60×10^{3}	6.00×10^{3}	0
g_{13}	5.89×10^{-1}	1.08×10^{0}	1.82×10^{0}	1.57×10^{0}	1.03×10^{1}

Table 14. Wilcoxon rank-sum test for the results of Table 13.

Comparison Criteria	**Compared Methods**	R^+	R^-	p-**Value**	**Best Method**
Average Errors	EDA-SPRS, EDA-MMSS	23	68	0.3560	–
	EDA-SPRS, DE/rand/1	19	72	0.0483	EDA-SPRS
	EDA-SPRS, jDE	15	76	0.0513	–
	EDA-SPRS, GADS	20	71	0.0649	–
	EDA-SPRS, DERSFTS	10	81	0.0513	–
	EDA-SPRS, OBDE	19	72	0.0544	–
	EDA-SPRS, NADE	15	76	0.0578	–
	EDA-SPRS, MUDE	20	71	0.0812	–
	EDA-SPRS, MDE-DS	81	10	0.0077	MDE-DS
	EDA-MMSS, DE/rand/1	10	81	0.1008	–
	EDA-MMSS, jDE	10	81	0.1119	–
	EDA-MMSS, GADS	10	81	0.1239	–
	EDA-MMSS, DERSFTS	10	81	0.1008	–
	EDA-MMSS, OBDE	10	81	0.0812	–
	EDA-MMSS, NADE	6	85	0.1119	–
	EDA-MMSS, MUDE	6	85	0.1662	–
	EDA-MMSS, MDE-DS	84	7	0.0025	MDE-DS

7. Conclusions

In this paper, a modified method of Estimation of Distribution Algorithms for simulation-based optimization is proposed to find the global optimum or near-optimum of noisy objective functions. The proposed method is composed of combining a modified version of Estimation of Distribution

Algorithm with a new sampling technique. This technique is a variable-sample method, in which function transportation is used with small-size samples in order to reduce the large dispersion resulting from random variables. The proposed sampling technique helps the search method to perform better exploration and hence to reach near global optimal solutions. The obtained results indicate the promising performance of the proposed method versus some existing state-of-the-arts methods, especially in terms of robustness and processing time.

Author Contributions: Conceptualization, A.-R.H. and A.A.A.; methodology, A.-R.H., A.A.A., and A.E.A.-H.; software, A.A.A.; validation, A.-R.H. and A.A.A.; formal analysis, A.-R.H., A.A.A., and A.E.A.-H.; investigation, A.-R.H. and A.A.A.; resources, A.-R.H., A.A.A., and A.E.A.-H.; data creation, A.-R.H. and A.A.A.; writing—original draft preparation, A.-R.H. and A.A.A.; writing—review and editing, A.-R.H. and A.E.A.-H.; visualization, A.-R.H., A.A.A., and A.E.A.-H.; project administration, A.-R.H.; funding acquisition, A.-R.H.'All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the National Plan for Science, Technology, and Innovation (MAARIFAH)—King Abdulaziz City for Science and Technology—the Kingdom of Saudi Arabia, award number (13-INF544-10).

Acknowledgments: The authors would like to thank King Abdulaziz City for Science and Technology, the Kingdom of Saudi Arabia, for supporting project number (13-INF544-10).

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Hard Test Functions

Twenty-five hard test functions h_1-h_{25} are listed in Table A1 with their global minima and bounds. The reader is directed to [47,48] for more details on these functions.

Table A1. Hard test functions.

h	Function Name	Bounds	Global Min
h_1	Shifted sphere function	$[-100, 100]$	-450
h_2	Shifted Schwefel's function 1.2	$[-100, 100]$	-450
h_3	Shifted rotated high conditioned elliptic function	$[-100, 100]$	-450
h_4	Shifted Schwefel's function 1.2 with noise in fitness	$[-100, 100]$	-450
h_5	Schwefel's function 2.6 with global optimum on bounds	$[-100, 100]$	-310
h_6	Shifted Rosenbrock's function	$[-100, 100]$	390
h_7	Shifted rotated Griewank's function without bounds	$[0, 600]$	-180
h_8	Shifted rotated Ackley's function with global optimum on bounds	$[-32, 32]$	-140
h_9	Shifted Rastrigin's function	$[-5, 5]$	-330
h_{10}	Shifted rotated Rastrigin's function	$[-5, 5]$	-330
h_{11}	Shifted rotated Weierstrass function	$[-0.5, 0.5]$	90
h_{12}	Schwefel's function 2.13	$[-100, 100]$	-460
h_{13}	Expanded extended Griewank's + Rosenbrock's function	$[-3, 1]$	-130
h_{14}	Expanded rotated extended Scaffer's function	$[-100, 100]$	-300
h_{15}	Hybrid composition function	$[-5, 5]$	120
h_{16}	Rotated hybrid composition function	$[-5, 5]$	120
h_{17}	Rotated hybrid composition function with noise in fitness	$[-5, 5]$	120
h_{18}	Rotated hybrid composition function	$[-5, 5]$	10
h_{19}	Rotated hybrid composition function with narrow basin global optimum	$[-5, 5]$	10
h_{20}	Rotated hybrid composition function with global optimum on the bounds	$[-5, 5]$	10
h_{21}	Rotated hybrid composition function	$[-5, 5]$	360
h_{22}	Rotated hybrid composition function with high condition number matrix	$[-5, 5]$	360
h_{23}	Non-Continuous rotated hybrid composition function	$[-5, 5]$	360
h_{24}	Rotated hybrid composition function	$[-5, 5]$	260
h_{25}	Rotated hybrid composition function without bounds	$[2, 5]$	260

Appendix B. Classical Test Functions—Set A

Appendix B.1. Goldstein and Price Function

Definition: $f_1(\mathbf{x}) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 13x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)][30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 - 48x_2 - 36x_1x_2 + 27x_2^2)]$.
Search space: $-2 \leq x_i \leq 2,\ i = 1, 2$.
Global minimum: $\mathbf{x}^* = (0, -1);\ f_1(\mathbf{x}^*) = 3$.

Appendix B.2. Rosenbrock Function

Definition: $f_2(\mathbf{x}) = \sum_{i=1}^{4} \left(100\left(x_i^2 - x_{i+1}\right))^2 + \left((x_i - 1)^2\right)\right) + 1$.
Search space: $-10 \leq x_i \leq 10,\ i = 1, 2, \ldots, 5$.
Global minimum: $\mathbf{x}^* = (1, \ldots, 1),\ f_2(\mathbf{x}^*) = 1$.

Appendix B.3. Griewank Function

Definition: $f_3(\mathbf{x}) = \frac{1}{40}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 2$.
Search space: $-10 \leq x_i \leq 10,\ i = 1, 2$.
Global minimum: $\mathbf{x}^* = (0, 0),\ f_3(\mathbf{x}^*) = 1$.

Appendix B.4. Pinter Function

Definition: $f_4(\mathbf{x}) = \sum_{i=1}^{n} ix_i^2 + \sum_{i=1}^{n} 20i \sin^2(x_{i-1} \sin x_i - x_i + \sin x_{i+1}) + \sum_{i=1}^{n} i \log_{10}[1 + i(x_{i-1}^2 - 2x_i + 3x_{i+1} - \cos x_i + 1)^2]$, where $x_0 = x_n$ and $x_{n+1} = x_1$.
Search space: $-10 \leq x_i \leq 10,\ i = i = 1, 2, \ldots, 5$.
Global minimum: $\mathbf{x}^* = (0, \ldots, 0),\ f_4(\mathbf{x}^*) = 1$.

Appendix B.5. Modified Griewank Function

Definition: $f_5(\mathbf{x}) = \frac{1}{40}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) - \prod_{i=1}^{n} e^{-x_i^2} + 2$.
Search space: $-10 \leq x_i \leq 10,\ i = 1, 2$.
Global minimum: $\mathbf{x}^* = (0, 0),\ f_5(\mathbf{x}^*) = 1$.

Appendix B.6. Griewank function with non-Gaussian noise

Definition: $f_6(\mathbf{x}) = \frac{1}{40}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 2$.
The simulation noise is changed to the uniform distribution U(-17.32, 17.32)
Search space: $-10 \leq x_i \leq 10,\ i = 1, 2$.
Global minimum: $\mathbf{x}^* = (0, 0), sf_6(\mathbf{x}^*) = 1$.

Appendix B.7. Griewank function with (50D)

Definition: $f_7(\mathbf{x}) = \frac{1}{40}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 2$.
Search space: $-10 \leq x_i \leq 10,\ i = 1, 2, \ldots, 50$.
Global minimum: $\mathbf{x}^* = (0, \ldots, 0), , f_7(\mathbf{x}^*) = 1$.

Appendix C. Classical Test Functions—Set B

Appendix C.1. Ackley Function

Definition: $g_1(\mathbf{x}) = 20 + e - 20e^{-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{-\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)}$.
Search space: $-15 \leq x_i \leq 30,\ i = 1, 2, \ldots, n$.
Global minimum: $\mathbf{x}^* = (0, \ldots, 0);\ g_1(\mathbf{x}^*) = 0$.

Appendix C.2. Alpine Function

Definition: $g_2(\mathbf{x}) = \sum_{i=1}^{n} |x_i \sin x_i + 0.1x_i|$.
Search space: $-10 \leq x_i \leq 10,\ i = 1, 2, \ldots, n$.
Global minimum: $\mathbf{x}^* = (0, \ldots, 0);\ g_2(\mathbf{x}^*) = 0$.

Appendix C.3. Axis Parallel Function

Definition: $g_3(\mathbf{x}) = \sum_{i=1}^{n} ix_i^2$.
Search space: $-5.12 \leq x_i \leq 5.12,\ i = 1, 2, \ldots, n$.
Global minimum: $\mathbf{x}^* = (0, \ldots, 0);\ g_3(\mathbf{x}^*) = 0$.

Appendix C.4. DeJong Function

Definition: $g_4(\mathbf{x}) = \|\mathbf{x}\|^2$.
Search space: $-5.12 \leq x_i \leq 5.12,\ i = 1, 2, \ldots, n$.
Global minimum: $\mathbf{x}^* = (0, \ldots, 0);\ g_4(\mathbf{x}^*) = 0$.

Appendix C.5. Drop Wave Function

Definition: $g_5(\mathbf{x}) = -\frac{1+\cos 12\sqrt{\|\mathbf{x}\|^2}}{\frac{1}{2}\|\mathbf{x}\|^2+2}$.
Search space: $-5.12 \leq x_i \leq 5.12,\ i = 1, 2, \ldots, n$.
Global minimum: $\mathbf{x}^* = (0, \ldots, 0);\ g_5(\mathbf{x}^*) = 0$.

Appendix C.6. Griewank Function

Definition: $g_6(\mathbf{x}) = \frac{1}{40}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 2$.
Search space: $-600 \leq x_i \leq 600,\ i = 1, 2, \ldots, 50$.
Global minimum: $\mathbf{x}^* = (0, \ldots, 0),\ g_6(\mathbf{x}^*) = 1$.

Appendix C.7. Michalewicz Function

Definition: $g_7(\mathbf{x}) = -\sum_{i=1}^{2} \sin(x_i) \sin^{2m}\left(\frac{ix_i^2}{\pi}\right);\ m = 10$.
Search space: $0 \leq x_i \leq \pi,\ i = 1, 2, \ldots, n$.
Global minima: $g_7(\mathbf{x}^*) = -29.6309; n = 30$.

Appendix C.8. Moved Axis Function

Definition: $g_8(\mathbf{x}) = \sum_{i=1}^{n} 5ix_i^2$.
Search space: $-5.12 \leq x_i \leq 5.12,\ i = 1, 2, \ldots, n$.
Global minimum: $\mathbf{x}^* = (0, \ldots, 0);\ g_8(\mathbf{x}^*) = 0$.

Appendix C.9. Pathological Function

Definition: $g_9(\mathbf{x}) = \sum_{i=1}^{n-1} \left[\frac{1}{2} + \frac{\sin^2\left(\sqrt{100x_i^2+x_{i+1}^2}-0.5\right)}{1+10^{-3}\left(x_i^2-2x_ix_{i+1}+x_{i+1}^2\right)^2}\right]$.
Search space: $-100 \leq x_i \leq 100,\ i = 1, 2, \ldots, n$.

Appendix C.10. Rastrigin Function

Definition: $g_{10}(\mathbf{x}) = 10n + \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i)\right)$.
Search space: $-2.56 \leq x_i \leq 5.12,\ i = 1, \ldots, n$.
Global minimum: $\mathbf{x}^* = (0, \ldots, 0),\ g_{10}(\mathbf{x}^*) = 0$.

Appendix C.11. Rosenbrock Function

Definition: $g_{11}(\mathbf{x}) = \sum_{i=1}^{4} \left(100\left(x_i^2 - x_{i+1}\right))^2 + \left((x_i - 1)^2\right) + 1$.
Search space: $-10 \leq x_i \leq 10,\ i = 1, 2, \ldots, 5$.
Global minimum: $\mathbf{x}^* = (1, \ldots, 1)$, $g_{11}(\mathbf{x}^*) = 1$.

Appendix C.12. Schwefel Function

Definition: $g_{12}(\mathbf{x}) = -\sum_{i=1}^{n} \left(x_i \sin\sqrt{|x_i|}\right)$.
Search space: $-500 \leq x_i \leq 500,\ i = 1, 2, \ldots, n$.
Global minimum: $\mathbf{x}^* = (1, \ldots, 1)$, $g_{12}(\mathbf{x}^*) = -418.9829n$.

Appendix C.13. Tirronen Function

Definition: $g_{13}(\mathbf{x}) = 3e^{-\frac{\|x\|^2}{10n}} - 10e^{-8\|x\|^2} + \frac{5}{2n}\sum_{i=1}^{n} \cos\left[5\left(x_i + (1 + i \ mod\ 2)\cos\|x\|^2\right)\right]$.
Search space: $-10 \leq x_i \leq 5,\ i = 1, 2, \ldots, n$.

References

1. Kizhakke Kodakkattu, S.; Nair, P. Design optimization of helicopter rotor using kriging. *Aircr. Eng. Aerosp. Technol.* **2018**, *90*, 937–945. [CrossRef]
2. Kim, P.; Ding, Y. Optimal design of fixture layout in multistation assembly processes. *IEEE Trans. Autom. Sci. Eng.* **2004**, *1*, 133–145. [CrossRef]
3. Kleijnen, J.P. Simulation-optimization via Kriging and bootstrapping: A survey. *J. Simul.* **2014**, *8*, 241–250. [CrossRef]
4. Fu, M.C.; Hu, J.Q. Sensitivity analysis for Monte Carlo simulation of option pricing. *Probab. Eng. Inf. Sci.* **1995**, *9*, 417–446. [CrossRef]
5. Plambeck, E.L.; Fu, B.R.; Robinson, S.M.; Suri, R. Throughput optimization in tandem production lines via nonsmooth programming. In Proceedings of the 1993 Summer Computer Simulation Conference, Boston, MA, USA, 19–21 July 1993; pp. 70–75.
6. Pourhassan, M.R.; Raissi, S. An integrated simulation-based optimization technique for multi-objective dynamic facility layout problem. *J. Ind. Inf. Integr.* **2017**, *8*, 49–58. [CrossRef]
7. Semini, M.; Fauske, H.; Strandhagen, J.O. Applications of discrete-event simulation to support manufacturing logistics decision-making: a survey. In Proceedings of the 38th Conference on Winter Simulation, Monterey, CA, USA, 3–6 December 2006; pp. 1946–1953.
8. Chong, L.; Osorio, C. A simulation-based optimization algorithm for dynamic large-scale urban transportation problems. *Transp. Sci.* **2017**, *52*, 637–656. [CrossRef]
9. Gürkan, G.; Yonca Özge, A.; Robinson, S.M. Sample-path solution of stochastic variational inequalities. *Math. Program.* **1999**, *84*, 313–333. [CrossRef]
10. Frühwirth-Schnatter, S. Data augmentation and dynamic linear models. *J. Time Ser. Anal.* **1994**, *15*, 183–202. [CrossRef]
11. Van Dyk, D.A.; Meng, X.L. The art of data augmentation. *J. Comput. Graph. Stat.* **2001**, *10*, 1–50. [CrossRef]
12. Andradóttir, S. Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice. Available online: https://www.wiley.com/en-us/Handbook+of+Simulation%3A+Principles%2C+Methodology%2C+Advances%2C+Applications%2C+and+Practice-p-9780471134039 (accessed on 16 March 2020).
13. Gosavi, A. Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning. Available online: https://www.researchgate.net/publication/238319435_Simulation-Based_Optimization_Parametric_Optimization_Techniques_and_Reinforcement_Learning (accessed on 16 March 2020).
14. Fu, M.C. Optimization for simulation: Theory vs. practice. *INFORMS J. Comput.* **2002**, *14*, 192–215. [CrossRef]
15. BoussaïD, I.; Lepagnot, J.; Siarry, P. A survey on optimization metaheuristics. *Inf. Sci.* **2013**, *237*, 82–117. [CrossRef]

16. Glover, F.W.; Kochenberger, G.A. *Handbook of Metaheuristics*; Springer: Boston, MA, USA, 2006.
17. Ribeiro, C.C.; Hansen, P. *Essays and surveys in metaheuristics*; Springer Science & Business Media: New York, NY, USA, 2012.
18. Siarry, P. Metaheuristics. Available online: https://link.springer.com/book/10.1007/978-3-319-45403-0#about (accessed on 18 March 2020).
19. Pellerin, R.; Perrier, N.; Berthaut, F. A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. *Eur. J. Oper. Res.* **2019**, *27*, 437. [CrossRef]
20. Mohamed, A.W.; Hadi, A.A.; Mohamed, A.K. Gaining-sharing knowledge based algorithm for solving optimization problems: A novel nature-inspired algorithm. *Int. J. Mach. Learn. Cybern.* **2019**, 1–29. [CrossRef]
21. Doğan, B.; Ölmez, T. A new metaheuristic for numerical function optimization: Vortex Search algorithm. *Inf. Sci.* **2015**, *293*, 125–145. [CrossRef]
22. Huang, C.; Li, Y.; Yao, X. A Survey of Automatic Parameter Tuning Methods for Metaheuristics. Available online: https://ieeexplore.ieee.org/document/8733017 (accessed on 16 March 2020).
23. Wang, J.; Zhang, Q.; Abdel-Rahman, H.; Abdel-Monem, M.I. A rough set approach to feature selection based on scatter search metaheuristic. *J. Syst. Sci. Complex.* **2014**, *27*, 157–168. [CrossRef]
24. Hedar, A.; Ali, A.F.; Hassan, T. Genetic algorithm and tabu search based methods for molecular 3D-structure prediction. *Numer. Algebra Control. Optim.* **2011**, *1*, 191–209. [CrossRef]
25. Hedar, A.; Fukushima, M. Heuristic pattern search and its hybridization with simulated annealing for nonlinear global optimization. *Optim. Methods Softw.* **2004**, *19*, 291–308. [CrossRef]
26. Hedar, A.; Fukushima, M. Directed evolutionary programming: To wards an improved performance of evolutionary programming. In Proceedings of the IEEE International Conference on Evolutionary Computation, Vancouver, BC, Canada, 16–21 July 2006; pp. 1521–1528.
27. Hedar, A.; Fukushima, M. Derivative-free filter simulated annealing method for constrained continuous global optimization. *J. Glob. Optim.* **2006**, *35*, 521–549. [CrossRef]
28. Larrañaga, P.; Lozano, J.A. *Estimation of Distribution Algorithms: A New Tool For Evolutionary Computation*; Springer Science & Business Media: Boston, MA, USA, 2001.
29. Hauschild, M.; Pelikan, M. An introduction and survey of estimation of distribution algorithms. *Swarm Evol. Comput.* **2011**, *1*, 111–128. [CrossRef]
30. Yang, Q.; Chen, W.N.; Li, Y.; Chen, C.P.; Xu, X.M.; Zhang, J. Multimodal estimation of distribution algorithms. *IEEE Trans. Cybern.* **2016**, *47*, 636–650. [CrossRef] [PubMed]
31. Krejca, M.S.; Witt, C. Theory of Estimation-of-Distribution Algorithms. Available online: https://www.researchgate.net/publication/337425690_Theory_of_Estimation-of-Distribution_Algorithms (accessed on 16 March 2020).
32. Homem-De-Mello, T. Variable-sample methods for stochastic optimization. *ACM Trans. Model. Comput. Simul. (Tomacs)* **2003**, *13*, 108–133. [CrossRef]
33. Faris, H.; Aljarah, I.; Mirjalili, S. Improved monarch butterfly optimization for unconstrained global search and neural network training. *Appl. Intell.* **2018**, *48*, 445–464. [CrossRef]
34. Lozano, J.A.; Larrañaga, P.; Inza, I.; Bengoetxea, E. Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms. Available online: https://link.springer.com/book/10.1007/3-540-32494-1#about (accessed on 18 March 2020).
35. Baluja, S. Population-Based Incremental Learning. A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning. Available online: https://www.ri.cmu.edu/pub_files/pub1/baluja_shumeet_1994_2/baluja_shumeet_1994_2.pdf (accessed on 16 March 2020).
36. Mühlenbein, H.; Paass, G. From Recombination of Genes to the Estimation Of Distributions I. Binary Parameters. Available online: http://www.muehlenbein.org/estbin96.pdf (accessed on 16 March 2020).
37. Pelikan, M.; Goldberg, D.E.; Lobo, F.G. A survey of optimization by building and using probabilistic models. *Comput. Optim. Appl.* **2002**, *21*, 5–20. [CrossRef]
38. Dong, W.; Wang, Y.; Zhou, M. A latent space-based estimation of distribution algorithm for large-scale global optimization. *Soft Comput.* **2018**, *8*, 1–23. [CrossRef]
39. Sebag, M.; Ducoulombier, A. Extending Population-Based Incremental Learning To Continuous Search Spaces. Available online: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.1884 (accessed on 16 March 2020).

40. Tsutsui, S.; Pelikan, M.; Goldberg, D.E. Evolutionary Algorithm Using Marginal Histogram Models in Continuous Domain. Available online: http://medal-lab.org/files/2001019.pdf (accessed on 16 March 2020).
41. Larranaga, P.; Etxeberria, R.; Lozano, J.; Pena, J.; Pe, J. Optimization by Learning and Simulation of Bayesian and Gaussian Networks. Available online: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.41.1895 (accessed on: 16 March 2020).
42. Larrañaga, P.; Etxeberria, R.; Lozano, J.A.; Peña, J.M. Optimization in cOntinuous Domains by Learning and Simulation of Gaussian Networks. Available online: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.31.3105 (accessed on: 16 March 2020).
43. Wang, J.; Hedar, A.; Zheng, G.; Wang, S. Scatter search for rough set attribute reduction. In Proceedings of the International Joint Conference on Computational Sciences and Optimization, Sanya, China, 24–26 April 2009; pp. 531–535.
44. Hedar, A.; Ali, A.F. Genetic algorithm with population partitioning and space reduction for high dimensional problem. In Proceedings of the International Conference on Computer Engineering & Systems, Cairo, Egypt, 14–16 December 2009; pp. 151–156.
45. Hedar, A.; Fukushima, M. Minimizing multimodal functions by simplex coding genetic algorithm. *Optim. Methods Softw.* **2003**, *18*, 265–282. [CrossRef]
46. Floudas, C.A.; Pardalos, P.M.; Adjiman, C.; Esposito, W.R.; Gümüs, Z.H.; Harding, S.T.; Klepeis, J.L.; Meyer, C.A.; Schweiger, C.A. *Handbook of Test Problems in Local and Global Optimization*; Springer Science & Business Media: Boston, MA, USA, 2013.
47. Liang, J.J.; Suganthan, P.N.; Deb, K. Novel composition test functions for numerical global optimization. In Proceedings of the 2005 IEEE Swarm Intelligence Symposium, Pasadena, CA, USA, 8–10 June 2005.
48. Suganthan, P.N.; Hansen, N.; Liang, J.J.; Deb, K.; Chen, Y.-P.; and Auger, A.; Tiwari, S. Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization. Available online: http://www.cmap.polytechnique.fr/~nikolaus.hansen/Tech-Report-May-30-05.pdf (accessed on 16 March 2020).
49. He, D.; Lee, L.H.; Chen, C.H.; Fu, M.C.; Wasserkrug, S. Simulation optimization using the cross-entropy method with optimal computing budget allocation. *ACM Trans. Model. Comput. Simul.* **2010**, *20*, 4. [CrossRef]
50. Hedar, A.; Allam, A.A. Scatter Search for Simulation-Based Optimization. In Proceedings of the 2017 International Conference on Computer and Applications, Dubai, UAE, 6–7 September 2017; pp. 244–251.
51. Hedar, A.; Allam, A.A.; Deabes, W. Memory-Based Evolutionary Algorithms for Nonlinear and Stochastic Programming Problems. *Mathematics* **2019**, *7*, 1126. [CrossRef]
52. García, S.; Fernández, A.; Luengo, J.; Herrera, F. A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. *Soft Comput.* **2009**, *13*, 959. [CrossRef]
53. Sheskin, D.J. *Handbook of Parametric and Nonparametric Statistical Procedures*; The CRC Press: Boca Raton, FL, USA, 2003.
54. Zar, J.H. *Biostatistical Analysis*; Pearson: London, UK, 2013.
55. Derrac, J.; García, S.; Molina, D.; Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol. Comput.* **2011**, *1*, 3–18. [CrossRef]
56. García-Martínez, S.; Molina, D.; Lozano, M.; Herrera, F. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behavior: a case study on the CEC'2005 special session on real parameter optimization. *J. Heuristics* **2009**, *15*, 617–644. [CrossRef]
57. Fan, Q.; Yan, X.; Xue, Y. Prior knowledge guided differential evolution. *Soft Comput.* **2017**, *21*, 6841–6858. [CrossRef]
58. Gosh, A.; Das, S.; Mallipeddi, R.; Das, A.K.; Dash, S.S. A modified differential evolution with distance-based selection for continuous optimization in the presence of noise. *IEEE Access* **2017**, *5*, 26944–26964. [CrossRef]