Memory-Based Evolutionary Algorithms for Nonlinear and Stochastic Programming Problems

Abdel-Rahman Hedar\textsuperscript{1,2,*}, Amira A. Allam\textsuperscript{1,3} and Wael Deabes\textsuperscript{1,4}\textsuperscript{\textdagger}

\textsuperscript{1} Department of Computer Science in Jamoum, Umm Al-Qura University, Makkah 25371, Saudi Arabia; wadeabes@uqu.edu.sa
\textsuperscript{2} Department of Computer Science, Faculty of Comp. & Info, Assiut University, Assiut 71526, Egypt
\textsuperscript{3} Department of Mathematics, Faculty of Science, Assiut University, Assiut 71516, Egypt; amirahm@science.au.edu.eg
\textsuperscript{4} Computers and Systems Engineering Department, Mansoura University, Mansoura 35516, Egypt
\textsuperscript{*} Correspondence: ahahmed@uqu.edu.sa or hedar@aun.edu.eg; Tel.: +966-55-0086-411 or +20-10-0070-4940

Received: 18 September 2019; Accepted: 7 November 2019; Published: 17 November 2019

Abstract: In this paper, we target the problems of finding a global minimum of nonlinear and stochastic programming problems. To solve this type of problem, we propose new approaches based on combining direct search methods with Evolution Strategies (ESs) and Scatter Search (SS) metaheuristics approaches. First, we suggest new designs of ESs and SS with a memory-based element called Gene Matrix (GM) to deal with those type of problems. These methods are called Directed Evolution Strategies (DES) and Directed Scatter Search (DSS), respectively, and they are able to search for a global minima. Moreover, a faster convergence can be achieved by accelerating the evolutionary search process using GM, and in the final stage we apply the Nelder-Mead algorithm to find the global minimum from the solutions found so far. Then, the variable-sample method is invoked in the DES and DSS to compose new stochastic programming techniques. Extensive numerical experiments have been applied on some well-known functions to test the performance of the proposed methods.

Keywords: global optimization; stochastic programming; simulation-based optimization; evolution strategies; scatter search; variable-sample techniques

1. Introduction

In the real world, there are many challenging applications which naturally require involving optimization techniques to have the best configurations, and find the optimal resources [1]. In particular, these complex application problems are computationally expensive in a sense of resources and time and most of them contain noise. Furthermore, the objective functions associated with most of these problems are not differentiable and have a large scale of parameters. Therefore, trials to compute the derivatives using finite differences methods mostly fail.

Typically, such real complex systems are usually modeled using a series of simulation procedures to evaluate their output responses. Also, they are integrated with an optimization algorithm to find their optimal parameters. The integration of the stochastic optimization methods into simulation software packages to meet their design requirements is crucial. Thus, these techniques are called Simulation-Based Optimization (SBO) [2,3]. SBO consists of three components, design tools, precise soft simulators, and optimization techniques. There are many engineering fields using the SBO, such as aerospace [4–6], automotive [7,8], transportation [9], and naval [10]. One of the most difficult issue of such systems is that the objective function usually contain noise and the user should carry out several simulations to obtain an acceptable approximate value of the objective function at a single point [11].
Therefore, this research focuses on optimizing objective functions with high noise which is known as stochastic programming \[12\].

Usually, the term “Simulation-Based Optimization” is more common than “stochastic optimization” \[11,13\]. Highly non-linear and/or high dimensional real-world optimization problems require the SBO, since these problems involve uncertainties in the form of randomness. This problem is known as stochastic programming problem where the estimation of the uncertainty is carried out as a stochastic probability function. The considered problem is represented mathematically as follows:

\[
\min_{x \in X} \{ f(x) = \mathbb{E}[\varphi(x, \omega)] \},
\]

where \( X \subseteq \mathbb{R}^n \) is the search space and \( f \) is a function with decision variables \( x \in X \), and \( \varphi \) is the probability function of a random variable \( \omega \). The configuration of the simulation parameters to enhance the operation is the main challenge. Typically, the simulation process is complex, since the evaluation of its objective function is computationally expensive, the results are noisy, and the objective function does not have an exact gradient. Usually, these problems have multiple local optima, which prevents classical nonlinear programming techniques from solving such problems successfully all the time. Therefore, global search methods are the best candidates to solve these problems which can defined in the following nonlinear programming problem:

\[
\min_{x \in X} f(x),
\]

where \( X \subseteq \mathbb{R}^n \) is the search space and \( f \) is a nonlinear function with decision variables \( x \in X \). Therefore, this research aims to design new methods to deal with Problems (1) and (2).

Recently the metaheuristics techniques, such as Evolution Strategies (ESs) and Scatter Search (SS), can solve Problem (2) by applying intelligent and learned procedures \[14–17\]. Metaheuristics are powerful tools since they are robust and can fight to successfully find accurate solutions of a wide range of problem areas \[14,17\]. However, applying the metaheuristics for complex problems converges slowly since exploring the global search space is a stochastic process that needs a high computational cost. Additionally, they usually could not remark promising search directions utilizing local information. On the other hand, the convergence of local search methods is faster since they create mathematical or logical movements based on the local information to determine promising search directions. However, they can be easily entrapped in local minima.

Typically, numerical global optimization methods are classified into deterministic, stochastic (heuristic), and hybrid techniques \[18,19\]. Generally, deterministic methods solve problems that are formulated based on their analytical properties, and some theoretical assumptions. Also, when there is available prior information about the problem under investigation, Branch-and-Bound algorithms in the deterministic global optimization are the most capable methods for searching the global minima \[20\]. The deterministic methods can find a global minimum with a pre-defined tolerance in case of having exact computation over long run time. In this type of optimization, the range of all variables defines the search space. Besides, these methods are characterized by having a theoretical assurance of finding the global solution. This means the value of the objective function when applying a local optimum differs just by small value from the global one. Old deterministic algorithms suffer from some limitations such as loose bounds, and they cannot deal with practical problems with more than ten variables. However, some new deterministic methods have been improved to obtain good results, such as the Hit-and-Run method. Also, it is modified to deal with high dimensional problems such as the Molecular Distance Geometry Problem (MDGP) instances \[21\], but it consumes high computational time. These limitations of the deterministic methods have motivated the transition to the stochastic methods.

On the other hand, some of stochastic methods are characterized by a probabilistic convergence guarantee, which usually means the probability of its global solution is one within infinite time.
Furthermore, the stochastic methods search for the optimal solution of problems that have black-box formulations, awful behavior, and noisy evaluation functions [22]. The stochastic methods are the best candidate to search for optimal solutions when objective functions suffer from a deficiency of prior suppositions. These methods work by sampling the search space randomly. There are many techniques classified as stochastic methods such as Genetic Algorithms (GA), Simulated Annealing (SA), Tabu Search (TS), Particle Swarm Optimization (PSO), and others [23]. Typically, the stochastic search is a multistage method which consists of global and local stages. The performance of these two stages significantly affects the efficiency of the stochastic methods, which also subject to the quality of sampled points.

Each technique in the above two categories has certain advantages and drawbacks. Researchers have hybridized these techniques to enhance their performance, and make it faster and more robust [24,25]. An effective introduction to hybrid metaheuristic methods is explained in [26]. An example for these hybrid algorithm is proposed in [27], where an approximation algorithm and linear relaxations can be integrated with an ant colony-like algorithm to get a very effective and fast algorithm for multi-period capacity and routing path planning in network design. Another example of work which combines genetic algorithms with individual Quasi-Newton learning for global optimizations is shown in [28].

The considered stochastic problem is conquered by several optimizations and search techniques, such as the Variable-Sample methods (VS) [29]. These methods are classified as Monte-Carlo methods that convert unconstrained stochastic optimization problems to deterministic problems. Explicitly, an estimation of the sample performance measure containing the random variable is carried out using several sample points. The VS method is invoked by a random search algorithm called Sampling Pure Random Search (SPRS) where an average approximation method with a variable-sample size replaces the objective function in each iteration [29].

In this paper, we find the global solution of the stochastic programming problem by developing new hybrid methods combining metaheuristics techniques, such as Evolution Strategies, Scatter Search, direct search, and Variable-Sample (VS) methods. Uncertainties formed as randomness and high dimensional are the main difficulties associated with such stochastic programming problems. Two main steps are representing the proposed methods as follows:

- First, new versions of ES and SS are proposed to enhance their performance in finding the solutions of the global optimization problem defined as Problem (2):
  - A new version of ESs called Direct Evolution Strategy (DES) [30] is applied. The DES method is composed by integrating the ESs with a new philosophy for applying the evolution process. Specifically, reproducing offspring in the DES method has two directions. The first one is to modify the mutation operation of ESs to reproduce offspring in a closed environment of their parents and use the parents’ experience to direct the offspring towards promising directions. The other direction of reproducing offspring is achieved by modifying the recombination operation to reproduce offspring in a promising area slightly away from their parents’ environment. This offspring has exploration duties that keep the diversity of its generation and the successive ones. This process of exploration is known in the literature as “Dispersal”, which means the movement of organisms away from their point of origin [31]. More elements have been added in the DES method to achieve more diversification process and to overcome the drawback of ESs termination.
  - A new SS method is proposed by extending the regular SS method built based on Laguna and Marti [32] and Duarte et al. [33] and integrating it with the Gene Matrix (GM). The proposed method is called Directed Scatter Search (DSS). Particularly, a hybrid technique is applied to update the reference set of the SS, where the possible values of each variable are divided into sub-ranges stored in the GM [34]. The GM enhances the exploration process after applying a
mutagenesis operator [34] by adding new diverse solutions to the search space. Mutagenesis operator works on accelerating the exploration process by altering some survival solutions.

- Second, the proposed metaheuristics techniques are combined with the Variable-Sample method to solve the simulation-based optimization problem defined as Problem (1):
  
  - The DES for Stochastic Programming (DESSP) method consists of a modified version of evolution strategies, and Monte-Carlo-based Variable-Sample technique to estimate the objective function values. The DESSP method tries to solve the stochastic global optimization problem. This problem has been considered in both cases as local and global problems.
  
  - A new method for dealing with the stochastic global optimization problem 1 is proposed. This method, called the Directed Scatter Search for Stochastic Programming (DSSSP), is an integration between the proposed SS method and the VS method. The DSSSP has two components. The Monte-Carlo-based variable-sample method is the first component that estimates the values of the objective function, while the DSS method is the second one generating solutions and exploring the search space.

The details of the proposed non-linear programming techniques and its main components are explained in Sections 2 and 3. The variable sample methods are highlighted in Section 4. The proposed simulation-based optimization methods are introduced in Sections 5 and 6. Section 7 explains the main numerical experiments and how the parameters of the proposed methods are selected and tuned. Finally, the conclusion is presented in Section 8.

2. Directed Evolution Strategies

Hereafter, we explain how the ESs method is modified to produce the Directed Evolution Strategies (DES) [30]. Then, the DES method is combined with a variable-sample technique to compose a simulation-based optimization method. Next, we mention some remarks about the ESs which considered as initiatives for building the DES.

In the \((\mu/\rho, \lambda)\)-ES, all parents are discarded during the selection process, whether they are good or bad since a strict Darwinian selection is the approved selection mechanism. This selection mechanism is not the optimal one since we lose some good solutions when discarding all of them. On the other hand, the \((\mu/\rho+\lambda)\)-ES applies different selection mechanism where the next generation only takes \(\mu\) which are the best individuals come from the parents and offspring. This type of selection may lead to immature convergence without guarantee diversity. Therefore, a new operation called “mutagenesis” is proposed. To maintain the diversity, the proposed mutagenesis alters some survival individuals to new unexplored areas. Besides, this new operation maintains the elitism works since it works within the “+” selection mechanism.

The ESs have an unbiased chance of reproducing children, which makes them are not like genetic algorithms (GAs). During recombination and mutation operations, there is an equal chance for all parents to reproduce children. Beside, the fittest works only during the selection operation. Moreover, the recombination operation in the standard ESs has a variable number \(\rho\) of mated parents, i.e., it supports the multi-parents recombination.

Since intelligent exploitation and exploration processes should be implemented in a good global search method, and since ESs have their promising mutation operation as an exploitation process, it is advisable to utilize a recombination mechanism which can serve as an intelligent exploration process. According to this concept, the recombination operation in the DES method has been constructed to diverse the search process towards new unexplored areas.

Another drawback in the ESs and EAs is termination criteria since there are no automatic termination criteria. In fact, in an early stage of the search process, the EAs may reach an optimal or near-optimal solution; however they could not terminate because they were not learned enough. Therefore, through successive generations, our proposed DES has termination measures to check the
progress of the exploration process. Afterwards, to converge faster an exploitation process refines the best candidates obtained so far.

The remarks mentioned above have been taken into consideration while designing the proposed DES. Next explained components are integrated into the standard ESs algorithm to create the new DES method.

- **Gene Matrix and Termination.** During the search process, $GM$ [34] is applied to ensure the exploration of the search space. In the $GM$, each variable is divided to sub-ranges based on its possible values. In the real-coding representation, there are $n$ variables or genes in each individual $x$ in the search space. The $GM$ checks the diversity by dividing the range of each gene to $m$ sub-ranges. Initially, a zero matrix of size the $n \times m$ represents the $GM$ where each element in each row denotes a sub-range of the equivalent gene. During the search process, $GM$’s values are altered from zeros to ones. An example of $GM$ in a two-dimension search space is shown in Figure 1. We achieve an advanced exploration process when the $GM$ has all ones. But the search will widely explore the recently chosen sub-ranges with the help of recombination operation. Therefore, the advantages of the $GM$ are the practical termination criteria and providing the search with diverse solutions as explained hereafter.

- **Mutation.** The standard ESs mutation process is applied for generating mutated children [35,36]. Thus, a mutated child $(\tilde{x}, \tilde{\sigma})$ is attained from a parent $(\hat{x}, \hat{\sigma})$, where the $i$-th component of the mutated child $(\tilde{x}, \tilde{\sigma})$ is given as

$$\tilde{\sigma}_i = \hat{\sigma}_i e^{\tau'N(0,1)+\tau N_i(0,1)},$$

$$\tilde{x}_i = \hat{x}_i + \tilde{\sigma}_i N_i(0,1),$$

where typically the values of $\tau' \propto 1/\sqrt{2\pi}$ and $\tau \propto 1/\sqrt{2\sqrt{n}}$ are selected as one.

- **Mutagenesis.** For accelerating the exploration process, the $N_{iw}$ worst individuals selected for the next generation are reformed by a more artificial mutation operation called “mutagenesis”. Explicitly, a random zero-position $(i,j)$ in $GM$ is selected. This will explore the $j$-th partition of the variable $x_i$ range which has not been visited yet. Then, one of the selected individuals for mutagenesis will be changed by a random value for $x_i$ within this partition. The formal mutagenesis procedure is explained as follows in Algorithm 1:
Algorithm 1 Mutagenesis

1: procedure Mutagenesis($x, GM$)
2:     if GM is not full then
3:         Select a zero-component ($i, j$) in GM randomly.
4:         $x \leftarrow x_i = l_i + (j - r) \frac{u_i - l_i}{m}$, where $l_i, u_i$ are the lower and upper bound of the variable $x_i$, respectively, and $r$ is a random number from $(0, 1)$.
5:     end if
6:     else
7:         Return
8:     end if
9: end procedure

Recombination. The recombination operation is not applied for all parents, but with probability $p_r \in [0, 1)$, since it has an exploration tendency. Initially, $\rho$ parents $p_1, \ldots, p_\rho$ will be randomly chosen from the current generation. The number of the partitions of each recombined child $\hat{x}$ equals $\rho$ ($\rho \leq n$) where partitions are taken from parents. Figure 2 displays an instant of 4 parents $p_1, p_2, p_3$, and $p_4$ partitioned at the same positions into 4 partitions. Next, a recombined child is created by inheriting its four partitions from $p_3, p_1, p_4$ and $p_2$, respectively. The DES recombination operation is described in Algorithm 2.

![Figure 2. An example of the recombination operation.](image)

Algorithm 2 Recombination

1: procedure Recombination($p_1, \ldots, p_\rho$)
2:     if $\rho \leq n$ then
3:         Divide each parent into $\rho$ parts at the same positions, i.e., $p_j = [X_{1j} \ X_{2j} \ \ldots \ X_{\rho j}], j = 1, \ldots, \rho$.
4:         Reorder the index set $\{1, 2, \ldots, \rho\}$ randomly to be $\{o_1, o_2, \ldots, o_\rho\}$.
5:         Compose a new child $x = [X_{1o_1}^X \ X_{2o_1}^X \ \ldots \ X_{\rho o_1}^X]$.
6:     else
7:         Return
8:     end if
9: end procedure

It is important to mention that the DES recombination procedure, defined to support the DES exploration process, is built as a standard type of the "single point crossover" and "two point crossover" [37]. Specifically in the high dimensional problems, the GM overcomes the complexity of such problem by not saving the information of the diverse sub-ranges of separate variables. Hence, as shown in the example in Figure 3a, it is possible to have misguided termination of the exploration
process. However, this drawback, shown in Figure 3b, can be easily overcome by calling such recombination operation as in Procedure 2.

Figure 3. The role of recombination operation and GM.

- **Selection.** The same standard ESs “+” selection operation is applied for the DES to choose from $P_g$ some survival individuals and insert it into its next $P_{g+1}$. Though, to ensure diversity, the mutagenesis operator alters some of these individuals.

- **Intensification Process.** After having all children, DES applies a local search starting from two characteristic children to improve themselves. These two children are the one with the best computed solution till now, and the other child is chosen after computing the objective function of each parent and his child and select the child with the highest difference in the current generation. Therefore, these characteristic are called children “the best child” and “the most promising child” respectively. The DES behaves like a “Memetic Algorithm” [38] after applying this intensification process which accelerates the convergence [39,40]. For more intensification to enhance the best solutions found so far, we apply derivative-free local search method as an efficient local search method. At the last stage, this intensification process can enhance convergence of EAs and prevent the EAs roaming around the global solution.

Figure 4 shows the main steps of the DES method while its algorithm is explained as follows (Algorithm 3):
Algorithm 3 DES Algorithm

1: Initial Population. Set the population parameters \( \mu \) and \( \lambda \), and calculate \( \nu := \lambda / \mu \). Generate an initial population \( P_0 = \{ (x^i, \sigma^i, F(x^i)) \mid i = 1, \ldots, \mu \} \).

2: Initial Parameters. Select the recombination probability \( p_r \in [0, 1] \) and set values of \( \rho \), \( N_{\text{ew}} \), \( m \) and \( N_{\text{elite}} \). Set the gene matrix \( GM \) to be an empty matrix of size \( n \times m \).

3: Initialize the generation counter \( g = 0 \).

4: while the termination conditions are not met do

5: for \( j \leftarrow 1, \mu \) do

6: Recombination. Choose a random number \( \chi \in [0, 1] \).

7: if \( \chi > p_r \) then

8: Select a parent \( (\hat{x}^j, \hat{\sigma}^j) \) from \( P_g \).

9: end if

10: else

11: Select \( \rho \) parents \( p_1, \ldots, p^\rho \) from \( P_g \).

12: Compute the recombined child \( (\hat{x}^j, \hat{\sigma}^j) \) using Recombination Procedure (Algorithm 2).

13: end if

14: Mutation. Use Equations (3) and (4) to compute a mutated children \( (\tilde{x}^{jk}, \tilde{\sigma}^{jk}) \), \( k = 1, \ldots, \nu \).

15: Fitness. Evaluate the fitness function \( \tilde{F}^{jk} = F(\tilde{x}^{jk}) \), \( k = 1, \ldots, \nu \).

16: end for

17: Children Pool. Set \( \mathcal{C}_g = \{ (\tilde{x}^{jk}, \tilde{\sigma}^{jk}, \tilde{F}^{jk}) \mid j = 1, \ldots, \mu, k = 1, \ldots, \nu \} \), to contain all children.

18: Selection. Set the next generation \( P_{g+1} \) to contain the best \( \mu \) individuals from \( P_g \cup \mathcal{C}_g \).

19: Mutagenesis. Use Mutagenesis Procedure (Algorithm 1) to alter the \( N_{\text{ew}} \) worst solutions in \( P_{g+1} \).

20: Update the gene matrix \( GM \).

21: \( g \leftarrow g + 1 \).

22: end while

23: Intensification. Refine the \( N_{\text{elite}} \) best solutions using local search.

---

**Figure 4.** The structure of the DES method.
3. Scatter Search with Gene Matrix

Hereafter, we explain the integration between the Gene Matrix (GM) explained in section 2, and the standard Scatter Search (SS) to compose the DSS method. The proposed DSS method involves the same five steps of the SS as follows:

1. **Diversification Generation**: Generating diverse trial solutions.
2. **Improvement**: Applying local search to improve these trial solutions.
3. **Reference Set Update**: Updating the reference set containing the \( b \) best solutions found. The solutions are classified based on the quality or the diversity.
4. **Solution Subset Generation**: Dividing the reference set solutions into subsets to create combined solutions.
5. **Solution Combination**: Combining solutions in the obtained subsets to create new trial solutions.

In step 3, the Reference Set (RefSet) in the proposed DSS has three parts. These parts are updated to keep the elitism, the diversity, while the GM and mutagenesis concepts [34] explained in Section 2 are applied to update the last part. The indicator of attaining an advanced exploration process is having GM full by ones. Therefore, GM assists the diversification process by supplying the search with diverse solutions using the mutagenesis operator (Algorithm 1), and it is used as a practical termination criteria.

3.1. Updating the Reference Set

In the basic procedure of SS based on the evaluation of the objective function, a solution with high value replaces the one with the worst value for updating the reference set.

Also, another part in the RefSet is kept preserving the solution diversity [32]. The standard update procedure in the SS is a reformed technique called the Update with Gene Matrix (UpGM) in the proposed DSS method. This new update technique increases the diversity, the exploration, and exploitation processes. All solutions from the combination ones already in the RefSet make up the new offspring Pool, i.e., Pool = Offspring \( \cup \) RefSet. Algorithm 4 represents the formal steps for the process of updating the RefSet.

**Algorithm 4 Update with GM**

```plaintext
1: procedure UpGM
2: RefSet\_new ← []
3: Elite Update. RefSet\_new ← the best \( b_1 \) solution in Pool.
4: Diverse Solutions. From the remaining solutions in Pool, choose the most \( b_2 (= b - b_1) \) diverse solutions to those in RefSet\_new using the max-min criterion.
5: Diversity Update. Add the diverse solutions generated in Step 4 to RefSet\_new.
6: GM Update. Use Mutagenesis Procedure (Algorithm 1) to generate \( N_{\text{m}} (< b) \) solutions, and add them to RefSet\_new.
7: Return RefSet\_new
8: end procedure
```

3.2. Diversity Control

In the standard SS to avoid trapping in local minima, the reference set has to be checked to circumvent duplicate solutions. However, applying this method cannot guarantee the diversity of the \( b_1 \) superior solutions while initializing the RefSet. Instead, a strict diversity check using the max-min criterion is applied to ensure the diversity of the \( b_2 \) solutions. In the initial RefSet, the chosen \( b_1 \) solutions can be exposed to a minimum diversity test as follows: The best solutions in the initial population \( P \) are chosen to be \( x^1 \) in the RefSet. Afterwards, \( x^1 \) is removed from \( P \), and the next high-quality solution \( x \) in \( P \) is selected and added to the reference set only if it is far from \( x^1 \). Therefore,
the minimum distance between the current solutions in RefSet and the selected solution \( x \) controls the process of adding the next best solution in \( P \) at each step. This distance has to be at least as large as the threshold value \( \epsilon \) to add this best solution.

3.3. DSS Algorithm

The main steps of the DSS method are shown in Figure 5 while it is formalized as described in the following Algorithm 5.

**Algorithm 5** DSS Algorithm

1: \( P \leftarrow \emptyset \).
2: \textbf{repeat}
3: Generate a solution \( x \) using the Diversification Generation Method.
4: \textbf{if} \((x \notin P \cup \text{is not close to any member in } P) \textbf{ then,}\)
5: \( P = (P \cup x) \).
6: \textbf{end if}
7: \textbf{until} the size of \( P \) is equal to \( P_{\text{Size}} \).
8: Compute the fitness values for all members in \( P \).
9: Apply Update with GM Procedure (Algorithm 4) to generate an initial RefSet.
10: \textbf{while} the stopping criteria are not met \textbf{do}
11: Use the Subset Generation Method to create solution subsets of size two.
12: Generate the offspring using the Combination Method.
13: Refine the offspring using the Improvement Method and add them to Pool Set.
14: Use the Update with GM Procedure (Algorithm 4) to update the RefSet.
15: \textbf{end while}
16: Refine the \( N_{\text{elite}} \) best solutions using a local search method.

![Figure 5. The structure of the DSS method.](image-url)

In this method, we apply “fminsearch.m” which is a MATLAB function implementing the Nelder-Mead method. This function improves the intensification of the proposed method (Step 16 of Algorithm 5). In the experimental work, the maximum evaluations number of the objective function is used to terminate the search process.
4. Variable Sample Method

The Variable-Sample (VS) method is classified as a class of Monte-Carlo methods for dealing with unconstrained stochastic optimization problems. The stochastic optimization problem can be converted using the VS to a deterministic problem, where an estimation of the sample performance measure containing the random variable is carried out using several sample points. The VS method is usually called by the Sampling Pure Random Search (SPRS), which is a random search algorithm [29]. In the SPRS, a sample of average approximation which has a VS size scheme replaces the objective function. The SPRS formal algorithm is described in Algorithm 6 [29].

Algorithm 6 Sampling Pure Random Search (SPRS) Algorithm
1: Generate a solution \( x_0 \in X \) randomly, choose an initial sample size \( N_0 \), and set \( k := 1 \).
2: while the termination conditions are not satisfied do
3:   Generate a solution \( y \in X \) randomly.
4:   Generate a sample \( \omega^1_k, \ldots, \omega^{N_k}_k \).
5:   for \( i = 1 \) to \( N_k \) do
6:     Compute \( \bar{f}(x, \omega^i_k) \).
7:   end for
8:   Compute \( \hat{f}(x_k) = \frac{\bar{f}(x, \omega^1_k) + \cdots + \bar{f}(x, \omega^{N_k}_k)}{N_k} \).
9:   If \( \hat{f}(y) < \hat{f}(x_k) \), then set \( x_{k+1} := y \). Otherwise, set \( x_{k+1} := x_k \).
10:  Update \( N_k \), and set \( k := k + 1 \).
end while

5. Directed Evolution Strategies for Simulation-based Optimization

The DESSP method is a modified version of the DES method to handle the simulation-based global optimization problem. The modified version (DESSP) evaluates the fitness function values by employing the VS method (Algorithm 6). To illustrate how this can be done, Figures 4 and 6 show the main structures of the DES and DESSP methods, respectively. One can note that the main added components in the DESSP method are the fitness function evaluation using variable sample technique and updating the size of the sample points. The first added component about function evaluations is performed as in Step 4 of Algorithm 6 with sample size \( N \). The parameter is configured through the following procedure shown as Algorithm 7.

Algorithm 7 Sample Size Update
1: procedure Sample Size Update
2: \( \alpha = \frac{\|GM\|}{|GM|} \), where \( \alpha \) is the update ratio \( |GM| \) is the size of the GM, and \( |GM|_1 \) is the number of ones in GM
3: \( N \leftarrow \alpha N_{\text{max}} \)
4: if \( N < N_{\text{min}} \) then
5:   \( N \leftarrow N_{\text{min}} \)
6: end if
7: Return \( N_{\text{new}} \)
8: end procedure

In Procedure 7, if \( N_{\text{max}} \) is a value representing the status of getting a full GM, then sample size \( N \) is taken as a portion of \( N_{\text{max}} \). The lower and upper bounds of the sample size are a predefined value \( N_{\text{min}} \) and \( N_{\text{max}} \), respectively.
6. Scatter Search for Simulation-Based Optimization

The DSS adapted to handle the same optimization problem using the following VS method.

6.1. Sampling and Fitness Evaluation

The VS technique is applied, as in Step 4 of Algorithm 6, to evaluate the objective (fitness) function, since it contains a random variable. The sample size parameter and its update are the main challenges with the fitness evaluation process. In the DSSSP method, the GM status can guide the configuration of the sample size parameter as described Sample Size Update Procedure (Algorithm 7).

6.2. DSSSP Algorithm

The DSS is modified to a new version called DSSSP, which tries to find the global optima in stochastic programming problems. Integrating the VS method into DSS is the main modification, Algorithm 6, to compute the fitness function values. The main steps in the DSSSP method are shown in Figure 7 and explained in Algorithm 8.
Algorithm 8 DSSSP Algorithm
1: $P \leftarrow \emptyset$
2: repeat
3:   Generate a solution $x$ using the Diversification Generation Method.
4:   if ($x \notin P \cup \emptyset$ is not close to any member in $P$) then,
5:     $P = (P \cup x)$.
6: end if
7: until the size of $P$ is equal to $P_{\text{Size}}$.
8: Use variable sampling with size $N$ to compute the fitness values for all members in $P$.
9: while the stopping criteria are not met do
10:   Use the Subset Generation Method to create solution subsets of size two.
11:   Generate the offspring using the Combination Method.
12:   Refine the offspring using the Improvement Method and add them to Pool Set.
13:   Use the Update with GM Procedure (Algorithm 4) to update the RefSet.
14:   Update the variable sample size $N$ using Sample Size Update Procedure (Algorithm 7).
15: end while
16: Refine the $N_{\text{elite}}$ best solutions using a local search method.

Figure 7. The structure of the DSSSP method.

7. Numerical Experiments

The proposed methods have been implemented using MATLAB, and some benchmark test functions have been applied to evaluate their performance. In this section, the configuration and the main numerical results prove the efficiency of these methods are shown.

7.1. Test Functions

For the nonlinear programming problem, we have tested the proposed algorithms using two sets of well-known test functions Set A, and Set B. Table 1 contains Set A [41–43], while Set B is listed in Table 2. These two sets of functions have enough diverse characteristics to test different difficulties that arise in global optimization problems.
Table 1. Test functions for global optimization: Set A.

| No. | f  | Function Name  | n   | No. | f  | Function Name  | n   |
|-----|----|----------------|-----|-----|----|----------------|-----|
| 1   | RC | Branin RCOS    | 2   | 8   | RT | Rastrigin      | 10  |
| 2   | GP | GoldsteinPrice | 2   | 9   | R  | Rosenbrock     | 10  |
| 3   | R  | Rosenbrock     | 2   | 10  | RT | Rastrigin      | 20  |
| 4   | H3,4| Hartmann       | 3   | 11  | R  | Rosenbrock     | 20  |
| 5   | S4,7| Shekel         | 4   | 12  | PW | Powell         | 24  |
| 6   | P4,0.5| Perm          | 4   | 13  | DP | DixonPrice     | 25  |
| 7   | T6 | Trid           | 6   | 14  | AK | Ackley         | 30  |

Table 2. Test functions for global optimization: Set B.

| No. | Function Name  | f  | n   | No. | Function Name  | f  | n   |
|-----|----------------|----|-----|-----|----------------|----|-----|
| 1   | Branin RCOS    | RC | 2   | 2   | Bohachevsky    | B  | 2   |
| 2   | Easom          | ES | 2   | 4   | Goldstein Price| GP | 2   |
| 3   | Shubert        | SH | 2   | 6   | Beale          | BL | 2   |
| 4   | Booth          | BO | 2   | 8   | Matyas         | MT | 2   |
| 5   | Hump           | HM | 2   | 10  | Schwefel       | SC | 2   |
| 6   | Rosenbrock     | R  | 2   | 12  | Zakharov       | Z  | 2   |
| 7   | De Joung       | DJ | 3   | 14  | Hartmann       | H3,4| 3   |
| 8   | Colville       | CV | 4   | 16  | Shekel         | S4,5| 4   |
| 9   | Shekel         | S4,7| 4  | 18  | Shekel         | S4,10| 4  |
| 10  | Perm           | P  | 4   | 20  | Perm           | P4,0.5| 4  |
| 11  | Power Sum      | PS | 4   | 22  | Hartmann       | H6,4| 6   |
| 12  | Schwefel       | SC | 6   | 24  | Trid           | T6 | 6   |
| 13  | Trid           | T10| 10  | 26  | Rastrigin      | RT | 10  |
| 14  | Griewank       | G10| 10  | 28  | Sum Squares    | SS | 10  |
| 15  | Rosenbrock     | R10| 10  | 30  | Zakharov       | Z10| 10  |
| 16  | Rastrigin      | RT | 20  | 32  | Griewank       | G20| 20  |
| 17  | Sum Squares    | S5 | 20  | 34  | Rosenbrock     | R20| 20  |
| 18  | Rastrigin      | Z  | 20  | 36  | Rastrigin      | PW | 24  |
| 19  | DixonPrice     | DP | 25  | 38  | Levy           | L30| 30  |
| 20  | Sphere         | SR | 30  | 40  | Ackley         | AK | 30  |

For the stochastic programming problem, seven test problems [44] are formulated from four documented test functions shown in Table 3, while their mathematical definitions are given in Appendix A [34].

Table 3. Stochastic programming test functions.

| Function Name  | n   | Stochastic Variable Distribution |
|----------------|-----|----------------------------------|
| f1             | 2   | N(0, 10)                          |
| f2             | 5   | N(0, 10)                          |
| f3             | 2   | N(0, 10)                          |
| f4             | 5   | N(0, 10)                          |
| f5             | 2   | N(0, 10)                          |
| f6             | 2   | Uniform(−17, 17)                  |
| f7             | 50  | N(0, 10)                          |

7.2. Parameter Setting and Tuning

To have a complete description of the proposed algorithms, all setting parameters of the DES, DSS, DESSP, and DSSSP methods are discussed in this section. Tables 4 and 5 summarize these parameters, their definitions, and best values. The values stated in the literature are used to set some parameters. A tuning process is carried out to obtain standard settings, while the proper values of the other parameters have been set by running some preliminary numerical experiments. These standard settings do not as much as possible, depend on the problem.
Different termination criteria are used in comparisons to stop the proposed methods. First, the automatic termination criterion based on the GM is discussed. We define $\gamma$ ratio to be equal to the number of ones in GM divided by the total number of entities of GM. If $\gamma$ is equal to 1 or close to 1, then the method can learn that a wide exploration process has been achieved. Therefore, the algorithm can be stopped after satisfying the condition $\gamma \geq \gamma_{\text{max}}$, where $\gamma_{\text{max}}$ is equal to 1 or close to 1. Figure 8 shows the results of applying the GM termination on the DESSP and DSSSP methods. In this figure, we allowed the algorithm to continue after reaching the termination point in order to notice any possible improvement after that. It is clear from the figure that there is almost no significant improvement after the GM termination point. The values of $\gamma_{\text{max}}$ are set equal to 0.8 and 1 for the DESSP and DSSSP methods, respectively. The difference in $\gamma_{\text{max}}$ values is a result because the DESSP method uses the GM to update its iterate solution sets more than that in the DSSSP method. Therefore, the DSSSP method reaches a full GM faster than the DSSSP method. In comparisons, we terminated the proposed methods according to the same termination criterion used in the compared methods in order to get a fair comparison [45,46].

Table 4. DES and DESSP parameters.

| Parameter | Definition | Best Value |
|-----------|------------|------------|
| $\mu$     | The population size | 50         |
| $\lambda$ | The offspring size | 10$\mu$    |
| $\rho$    | No. of mated parents | $\min(n, 5)$ |
| $p_r$     | The recombination probability | 0.25       |
| $N_{\text{min}}$ | The initial value for the sample size | 100        |
| $N_{\text{max}}$ | The maximum value for the sample size | 5000       |
| $N_{\text{elite}}$ | No. of best solutions used in in the intensification step | 1          |
| $N_{\text{wr}}$ | No. of worst solutions updated by the mutagenesis process | $n$        |
| $m$       | No. of partitions in GM | 50         |

Figure 8. Automatic termination based on the GM.
Table 5. DSS and DSSP parameters.

| Parameter | Definition                                      | Best Value |
|-----------|------------------------------------------------|------------|
| Psize     | The initial population size                     | 100        |
| b         | The size of the RefSet                          | 10         |
| b1        | No. of elite solutions in RefSet                | 6          |
| b2        | No. of diverse solutions in RefSet              | 4          |
| Nmin      | The initial value for the sample size           | 100        |
| Nmax      | The maximum value for the sample size           | 5000       |
| Nelite    | No. of best solutions used in in the intensification step | 1 |
| Nw        | No. of worst solutions updated by the mutagenesis process | 3 |
| m         | No. of partitions in GM                         | 50         |

7.3. Numerical Results on Functions without Noise

This section contains the explanation of the experiments’ implementation that evaluates the proposed algorithms and show their results. Initially, two experiments are carried out to test the DSS algorithm. The first experiment measures the effect of applying Gene Matrix to the standard SS. The experimental results are reported after applying the SS algorithm with and without using GM on the Set A test functions stated in Table 1. On the other hand, the second one compares the performance of the DSS and the standard SS method as well as other metaheuristics methods.

The optimality gap of the above experiments is defined as:

\[ GAP = |f(x) - f(x^*)|, \]

where \( x \) and \( x^* \) are a heuristic solution, and the optimal solution, respectively. We consider \( x \) as a best solution if the average \( GAP \leq 0.001 \).

The results of the first experiment after updating RefSet with (DSS) and without using GM (standard SS) are reported in Table 6. These results are calculated after running each test function; ten independent runs with a maximum number of evaluating the objective function equals 20000 for each run. The results, shown in Table 6, illustrate both the average gap (Av. GAP) and the success rates (Suc.) of all functions. The results, especially for the higher dimensional problems, reveal that the DSS algorithm has steadily better performance than the standard SS in terms of Av. GAP and the ability to obtain a global solution.

Table 6. DSS and SS results on Set A.

| f          | SS     | DSS     |
|------------|--------|---------|
|            | Av. GAP | Suc.   | Av. GAP | Suc.   |
| RC         | 3.58x10^-7 | 100    | 3.581x10^-7 | 100    |
| GP         | 4.69x10^-11 | 100    | 5.35x10^-11 | 100    |
| R2         | 1.1x10^-11  | 100    | 2.56x10^-11 | 100    |
| H3,4       | 2.1x10^-6   | 100    | 2.15x10^-6  | 100    |
| S4,7       | 2.9x10^-7   | 100    | 3.22x10^-7  | 100    |
| P4,5       | 0.6        | 13     | 0.55     | 27     |
| T6         | 2.39x10^-3  | 80     | 2.703x10^-3 | 77     |
| RT10       | 4.66x10^-6  | 100    | 1.92x10^-6 | 100    |
| R10        | 18.59      | 0      | 15.01    | 0      |
| RT20       | 3.558      | 0      | 5.01     | 0      |
| R20        | 45.84      | 0      | 29.77    | 0      |
| PW24       | 47.1       | 0      | 16.2     | 0      |
| DP25       | 3.51       | 0      | 1.4      | 0      |
| AK30       | 10.39      | 0      | 8.28     | 0      |
The average gap calculated using DSS algorithm, with the maximum number of function evaluation equals 50,000, is reported in Table 7 over the set of 40 functions listed as Set B in Table 2. The DSS algorithm closely reaches near to global solutions as proven from the results shown in Tables 6 and 7.

### Table 7. DSS results on Set B.

| No. | \(f\) | \(\text{Av. GAB}\) | No. | \(f\) | \(\text{Av. GAB}\) |
|-----|------|-----------------|-----|------|-----------------|
| 1   | RC   | \(3.5773 \times 10^{-7}\) | 2   | \(B_2\) | 0               |
| 3   | ES   | 0               | 4   | \(G_2\) | \(7.7538 \times 10^{-14}\) |
| 5   | SH   | \(8.83102 \times 10^{-6}\) | 6   | \(BL\) | 0               |
| 7   | BO   | 0               | 8   | \(MT\) | \(1.73546 \times 10^{-66}\) |
| 9   | HM   | \(4.65101 \times 10^{-8}\) | 10  | \(SC_2\) | 2.54551          |
| 11  | \(R_2\) | 0               | 12  | \(Z_2\) | \(2.88141 \times 10^{-64}\) |
| 13  | \(D_{ij}\) | \(9.77667 \times 10^{-57}\) | 14  | \(H_{5,4}\) | 2.14782          |
| 15  | CV   | \(2.05 \times 10^{-4}\) | 16  | \(S_{4,5}\) | 3.41193          |
| 17  | \(S_{4,7}\) | \(5.66819 \times 10^{-7}\) | 18  | \(S_{4,10}\) | 9.81669          |
| 19  | \(P_{4,0.5}\) | \(3.83 \times 10^{-4}\) | 20  | \(P_{0,4,0.5}\) | 5.52274          |
| 21  | \(P_{8,18,44,114}\) | \(6.09758 \times 10^{-5}\) | 22  | \(H_{6,4}\) | 2.05919          |
| 23  | \(SC_6\) | 78.99516181    | 24  | \(T_6\) | 7.58971          |
| 25  | \(T_{10}\) | \(8.06413 \times 10^{-9}\) | 26  | \(RT_{10}\) | 1.91653          |
| 27  | \(G_{10}\) | 0.142472164   | 28  | \(S_{10}\) | 3.75918          |
| 29  | \(R_{10}\) | 0.707283407   | 30  | \(Z_{10}\) | 1.92175          |
| 31  | \(RT_{20}\) | 7.98934 \times 10^{-12} | 32  | \(G_{20}\) | 0.052065617      |
| 33  | \(S_{29}\) | 5.41513 \times 10^{-12} | 34  | \(R_{20}\) | 0.799395703      |
| 35  | \(Z_{20}\) | 0.19800033    | 36  | \(P_{24}\) | 8.33102 \times 10^{-5} |
| 37  | \(DP_{25}\) | 0.943358525   | 38  | \(L_{30}\) | 1.29657 \times 10^{-9} |
| 39  | \(SR_{30}\) | 1.26068 \times 10^{-11} | 40  | \(AK_{30}\) | 1.08842 \times 10^{-6} |

A performance comparison is carried out in the second experiment between the DSS algorithm and standard Scatter Search (SS) algorithms [32] as well as the other two metaheuristics methods designed for the continuous optimization problems. These other two methods are the Genetic Algorithm (GA) for constrained problems (Genocop III) [47], and Directed Tabu Search (DTS) [43]. The results of the Av. GAP calculated over the 40 functions of Set B at several points during the search are shown in Table 8. The method [32] ignores the GAP for Problem 23; therefore, we also ignore the result of that problem and provide the results of 39 problems.

### Table 8. The comparison results of DSS and other metaheuristics.

| Evaluations | Genocop III | SS  | DTS  | DSS  |
|------------|-------------|-----|------|------|
| 500        | 611.30      | 26.34 | 43.06 | 112.08 |
| 1000       | 379.03      | 14.66 | 24.26 | 56.24 |
| 5000       | 335.81      | 4.96  | 4.22  | 34.88 |
| 10,000     | 328.66      | 3.60  | 1.80  | 5.36  |
| 20,000     | 324.72      | 3.52  | 1.70  | 0.9031 |
| 50,000     | 321.20      | 3.46  | 1.29  | 0.0714 |

Our proposed DSS algorithm performs better than the SS as designed in [32], since the \(\text{GAP} < 1.0\); except problem 23 with \(\text{GAP} = 78.99516181\). All the SS have \(\text{GAP} < 1\), except problems 23, 26, 34, and 40, which have \(\text{GAP}\) values equal 118.4341, 9.9496, 2.2441, and 5.5033, respectively. Another comparison criteria is the number of solved problems. The DSS solves 33 problems after 50000 function evaluations, however the SS [32] just solves 30 problems even with increasing the number of the functions evaluations from 20000 up to 50000.

The results in Table 8 show a substantial improvement of the values of Av. GAP applying DSS algorithm compared to the other methods at a large number of function evaluation. These results
demonstrate the standing of the DSS method on the exploration process, which requests a high cost of function evaluations to guarantee good coverage to the search space. An additional performance measure is the number of optimal solutions found while applying each method.

Figure 9 illustrates the number of optimal solutions reached by each algorithm during the search process with a stopping condition of 50,000 objective function evaluations. The figure demonstrates a better performance of the DSS algorithm at a large number of function evaluations by solving 33 at 50,000 evaluations, while Genocop III, SS, and DTS successfully solve a total of 23, 30, and 32 problems, respectively.

**Figure 9. Number of solved problems.**

### 7.4. Stochastic Programming Results

The main results of the DESSP and DSSSP methods are shown in Table 9, which includes the averages of the best and the average of the error gaps of the obtained solutions. These results obtained through 100 independent runs of the DESSP and DSSSP codes with a maximum number of function evaluations equal to 500,000 in each run. These results illustrate that the DSSSP could attain very close solutions for four test functions $f_1, f_3, f_5$, and $f_6$. The results for functions $f_2$ and $f_4$ are not so far from the optimal ones. However, due to the big size of function $f_7$, the result of DESSP was not close to the optimal solution. Moreover, the DSSSP algorithm has revealed a best performance especially for high dimensional function $f_7$. For four test functions $f_1$ and $f_6$, the DSSSP can attain very close solutions, and the solutions for functions $f_2, f_3, f_4,$ and $f_5$ are near the optimal ones. The performance of the proposed methods is measured with more accurate metric called Normalized Distance [48], which is defined using three components as follows:

- **Normalized Euclidean Distance ($\Delta x$):**
  \[
  \Delta x = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left( \frac{\bar{x}_i - x^*_i}{U_i - L_i} \right)^2},
  \]

  where $x^* = (x^*_1, \ldots, x^*_n)$ is the exact solution, $\bar{x} = (\bar{x}_1, \ldots, \bar{x}_n)$ is the approximate solution obtained by the proposed methods, and $U = (U_1, \ldots, U_n), L = (L_1, \ldots, L_n)$ are upper and lower bounds of the search space, respectively.

- **Associate Normalized Distance ($\Delta f$):**
  \[
  \Delta f = \frac{\tilde{f}_{\text{min}} - f^*_\text{min}}{f^*_\text{max} - f^*_\text{min}},
  \]

  where $\tilde{f}_{\text{min}}$ is the best minimum value of the objective function obtained by the proposed methods, and $f^*_\text{max}, f^*_\text{min}$ are the global maximum and minimum values of the objective function, respectively, within the search space.
Total Normalized Distance ($\Delta_t$):

$$\Delta_t = \sqrt{\frac{\Delta x^2 + (\Delta f)^2}{2}}.$$ 

Table 10 shows the values of these metrics for the results obtained by the DESSP and DSSSP methods.

**Table 9.** Error gap results obtained by the DESSP and DSSSP methods.

| $f$ | DESSP | DSSSP |
|-----|-------|-------|
|     | Best  | Average | Best  | Average |
| $f_1$ | 0.0005 | 0.2326 | 0.0146 | 0.2939 |
| $f_2$ | 0.8046 | 3.5496 | 0.4080 | 6.5550 |
| $f_3$ | 0.0010 | 0.3313 | 0.5900 | 1.0400 |
| $f_4$ | 0.1442 | 3.7538 | 2.7487 | 6.7145 |
| $f_5$ | 0.0004 | 0.3871 | 0.2819 | 1.9135 |
| $f_6$ | 0.00001 | 0.0213 | 0.0003 | 0.0921 |
| $f_7$ | 27.8824 | 39.5636 | 8.4088 | 12.3794 |

**Table 10.** Averages of normalized Euclidean distances obtained by the DESSP and DSSSP methods.

| $f$ | DESSP | DSSSP |
|-----|-------|-------|
|     | $\Delta x$ | $\Delta f$ | $\Delta t$ | $\Delta x$ | $\Delta f$ | $\Delta t$ |
| $f_1$ | $8.1364 \times 10^{-4}$ | $2.2982 \times 10^{-7}$ | $5.7533 \times 10^{-4}$ | $4.1030 \times 10^{-3}$ | $2.9045 \times 10^{-7}$ | $2.9013 \times 10^{-3}$ |
| $f_2$ | $1.2902 \times 10^{-2}$ | $7.3332 \times 10^{-7}$ | $9.1231 \times 10^{-3}$ | $4.2088 \times 10^{-2}$ | $1.3542 \times 10^{-6}$ | $2.9761 \times 10^{-2}$ |
| $f_3$ | $1.3467 \times 10^{-2}$ | $5.0255 \times 10^{-2}$ | $3.8547 \times 10^{-2}$ | $7.9792 \times 10^{-2}$ | $1.5776 \times 10^{-1}$ | $1.2763 \times 10^{-1}$ |
| $f_4$ | $1.0589 \times 10^{-2}$ | $9.7511 \times 10^{-5}$ | $7.4884 \times 10^{-3}$ | $2.5053 \times 10^{-2}$ | $2.0036 \times 10^{-5}$ | $17.7155 \times 10^{-3}$ |
| $f_5$ | $1.0925 \times 10^{-2}$ | $5.0898 \times 10^{-2}$ | $3.8599 \times 10^{-2}$ | $6.6194 \times 10^{-2}$ | $2.5204 \times 10^{-1}$ | $1.8554 \times 10^{-1}$ |
| $f_6$ | $7.6039 \times 10^{-3}$ | $3.2361 \times 10^{-3}$ | $6.6166 \times 10^{-3}$ | $2.2119 \times 10^{-2}$ | $1.3976 \times 10^{-2}$ | $2.0627 \times 10^{-2}$ |
| $f_7$ | $1.6132 \times 10^{-1}$ | $1.6105 \times 10^{-1}$ | $1.6134 \times 10^{-1}$ | $2.7005 \times 10^{-1}$ | $5.0392 \times 10^{-2}$ | $1.9430 \times 10^{-1}$ |

Averages of processing time for running the DESSP and DSSSP codes are shown in Table 11. These averages are recorded in seconds after running the proposed codes on iMac machine with 3.2 GHz Quad-Core Intel Core i5 processor and 24 GB 1600 MHz DDR3 memory. The processing time of the DESSP code is better than that of the DSSSP code as shown in Table 11. However, the statistical test results in Table 12 do not favor either method against the other at the significance level 5%.

**Table 11.** Averages of processing time (in seconds) for running the DESSP and DSSSP codes.

| $f$ | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ | $f_6$ | $f_7$ |
|-----|-------|-------|-------|-------|-------|-------|-------|
| DESSP | 162 | 20 | 119 | 177 | 25 | 10 | 476 |
| DSSSP | 414 | 103 | 317 | 307 | 114 | 35 | 683 |

In order to clarify the performance difference between the proposed methods, we can use the Wilcoxon rank-sum test [49–51]. This test belongs to the class of non-parametric tests which are recommended in our experiments [52,53]. Table 12 summarizes the results of applying Wilcoxon test on the results obtained by the proposed methods, it displays the sum of rankings obtained in each comparison and the $p$-value associated. In this statistical test, we assume that $d_i$ is the difference between the performance scores of the two methods on $i$-th out of $N$ results. Then, we rank the differences according to their absolute values. The ranks $R^+$ and $R^−$ are computed as follows:

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

$$R^- = \sum_{d_i < 0} \text{rank}(d_i) + \frac{1}{2} \sum_{d_i = 0} \text{rank}(d_i).$$
The null hypothesis of equality of means is rejected if the \( \min(R^+, R^-) \) is less than or equal to the value of the distribution of Wilcoxon for \( N \) degrees of freedom (Table B.12 in [51]).

Although the shown results indicate that the DESSP method is slightly better than the DSSSP method, the statistical test results in Table 12 show that there is no significant difference between the two methods at the significance level 5%.

| Comparison Criteria   | \( R^- \) | \( R^+ \) | \( p \)-Value | Best Method |
|-----------------------|----------|----------|---------------|-------------|
| Best Solutions        | 11       | 17       | 0.3829        | –           |
| Average Solutions     | 7        | 21       | 0.6200        | –           |
| \( \Delta x \)        | 8        | 28       | 0.0973        | –           |
| \( \Delta f \)        | 6        | 22       | 0.7104        | –           |
| \( \Delta t \)        | 0        | 28       | 0.3176        | –           |
| Processing Time       | 0        | 28       | 0.2593        | –           |

Table 12. Wilcoxon rank-sum test of DESSP and DSSSP results.

The DESSP results have been compared against the results of the standard evolution strategies (ES) method and one of the best version of it which is called Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [54]. The comparison results are shown in Table 13, which includes the averages of the best function values obtained by each method within 100 independent runs. From Table 13, the DESSP method has shown superior performance against the compared methods.

Table 13. The averages of the best function values obtained by ES, CMA-ES, and DESSP.

| Method | \( f_1 \) | \( f_2 \) | \( f_3 \) | \( f_4 \) |
|--------|------------|------------|------------|------------|
| ES     | 47.7325    | \( 1.93 \times 10^4 \) | 3.265      | 970.7125   |
| CMA-ES | 13.2173    | 12.7998    | 2.4712     | 23.6619    |
| DESSP  | 3.0005     | 1.8046     | 1.0010     | 1.1442     |

A comparison between the proposed methods and the Cross-Entropy method [44] is also carried out. Figures 10 shows the results, which promise potentials of the proposed methods in attaining better objective function values in some tested function. It is worthwhile to mention that the final intensification plays an important role in improving the obtained solutions, as shown in the method performance with function \( f_1 \) in Figure 10.

![Figure 10. Cont.](image-url)
8. Conclusions

In this paper, novel algorithms are designed to search for a globally optimal solution or near-globally optimal solution of an objective function that having random noisy variables. The proposed methods combine a modified version of the scatter search and evolution strategies with a variable-sample method, which is a Monte-Carlo based to handle the random variables. Moreover, a memory element called gene matrix is invoked which could guide the search process to new diverse solutions and also help in terminating the algorithms. We have implemented these methods in MATLAB. The performance of these proposed methods is tested using several sets of benchmark test functions of nonlinear and stochastic programming problems. The obtained results proved that these novel methods have promising performance compared with some standard methods in the literature. As future work, many other metaheuristics can be adapted to deal with stochastic programming by invoking different sampling techniques in order to interface the problem of noise. In addition, the fuzzy and rough set theories can be modeled instead of sampling techniques to deal with the uncertainty of objective functions with noise.

Figure 10. Comparison using test functions $f_1$–$f_7$. 
Author Contributions: Conceptualization, A.H. and A.A.; methodology, A.H., A.A. and W.D.; software, A.A.; validation, A.H. and A.A.; formal analysis, A.H., A.A. and W.D.; investigation, A.H. and A.A.; resources, A.H., A.A. and W.D.; data creation, A.A. and W.D.; writing—original draft preparation, A.H. and A.A.; writing—review and editing, A.H. and W.D.; visualization, A.H. and A.A. and W.D.; project administration, A.H.; funding acquisition, A.H.

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).

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

Appendix A. Test Functions

Appendix A.1. Goldstein & Price Function

\[ f_1(x) = (1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 13x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)) \]
\[ * (30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 - 48x_2 - 36x_1x_2 + 27x_2^2)). \]

Search space: \[-2 \leq x_i \leq 2, \ i = 1, 2.\]
Global minimum: \( x^* = (0, -1); \ f_1(x^*) = 3. \)

Appendix A.2. Rosenbrock Function

\[ f_2(x) = \sum_{i=1}^{n-1} \left( 100 (x_i^2 - x_{i+1})^2 + ((x_i - 1)^2) \right) + 1. \]
Search space: \[-10 \leq x_i \leq 10, \ i = 1, \ldots, n.\]
Global minimum: \( x^* = (1, \ldots, 1), f_2(x^*) = 1. \)

Appendix A.3. Griewank Function

\[ f_3(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos \left( \frac{x_i}{\sqrt{n}} \right) + 2. \]
Search space: \[-10 \leq x_i \leq 10, \ i = 1, \ldots, n.\]
Global minimum: \( x^* = (0, 0), f_3(x^*) = 1. \)

Appendix A.4. Pinter Function

\[ f_4(x) = \sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n} 20i \sin^2(x_{i-1} \sin x_i - x_i + \sin x_{i+1}) \]
Search space: \[-10 \leq x_i \leq 10, \ i = 1, \ldots, n.\]
Global minimum: \( x^* = (0, \ldots, 0), f_4(x^*) = 1. \)

Appendix A.5. Modified Griewank Function

\[ f_5(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos \left( \frac{x_i}{\sqrt{n}} \right) - \prod_{i=1}^{n} e^{-x_i^2} + 2. \]
Search space: \[-10 \leq x_i \leq 10, \ i = 1, \ldots, n.\]
Global minimum: \( x^* = (0, 0), f_5(x^*) = 1. \)

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

\[ f_6(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos \left( \frac{x_i}{\sqrt{n}} \right) + 2. \]
The simulation noise is changed to the uniform distribution \( U(-17.32, 17.32) \).
Search space: \[-10 \leq x_i \leq 10, \ i = 1, \ldots, n.\]
Global minimum: \( x^* = (0, \ldots, 0), f_6(x^*) = 1. \)

Appendix A.7. Griewank Function (n = 50)

\[ f_7(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos \left( \frac{x_i}{\sqrt{n}} \right) + 2. \]
Search space: \[-10 \leq x_i \leq 10, \ i = 1, \ldots, n.\]
Global minimum: \( x^* = (0, \ldots, 0), 1f_7(x^*) = 1. \)
References

1. Osorio, C.; Selvama, K.K. Simulation-based optimization: Achieving computational efficiency through the use of multiple simulators. Transp. Sci. 2017, 51, 395–411. doi:10.1287/trsc.2016.0673.
2. Campana, E.F.; Diez, M.; Lemma, U.; Liuzzi, G.; Lucidi, S. Derivative-free global ship design optimization using global/local hybridization of the DIRECT algorithm. Optim. Eng. 2016, 17, 127–156. doi:10.1007/s11081-015-9303-0.
3. 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.
4. Simpson, T.W.; Mauery, T.M.; Korte, J.J.; Mistree, F. Kriging models for global approximation in simulation-based multidisciplinary design optimization. AIAA J. 2001, 39, 2233–2241. doi:10.2514/2.1234.
5. Schillings, C.; Schmidt, S.; Schulz, V. Computers & Fluids Efficient shape optimization for certain and uncertain aerodynamic design. Comput. Fluids 2011, 46, 78–87. doi:10.1016/j.compfluid.2010.12.007.
6. Hu, X.; Chen, X.; Parks, G.T.; Yao, W. Review of improved Monte Carlo methods in uncertainty-based design optimization for aerospace vehicles. Progress Aerospace Sci. 2016, 86, 20–27.
7. Hojjat, M.; Stavropoulou, E.; Bletzinger, K.U. The Vertex Morphing method for node-based shape optimization. Comput. Methods Appl. Mech. Eng. 2014, 268, 494–513. doi:10.1016/j.cma.2013.10.015.
8. Chong, L.; Osorio, C. A simulation-based optimization algorithm for dynamic large-scale urban transportation problems. Transp. Sci. 2017, 52, 637–656.
9. Mall, P.; Fidlin, A.; Krüger, A.; Groß, H. Simulation based optimization of torsional vibration dampers in automotive powertrains. Mech. Mach. Theory 2017, 115, 244–266.
10. Diez, M.; Volpi, S.; Serani, A.; Stern, F.; Campana, E.F. Simulation-Based Design Optimization by Sequential Multi-criterion Adaptive Sampling and Dynamic Radial Basis Functions; In Advances in Evolutionary and Deterministic Methods for Design, Optimization and Control in Engineering and Sciences, Springer International Publishing AG: Cham, Switzerland, 2019; Volume 48, pp. 213–228. doi:10.1007/978-3-319-89988-6_13.
11. Gosavi, A. Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning; Springer: New York, USA, 2015.
12. Birge, J.R.; Louveaux, F. Introduction to Stochastic Programming; Springer Science & Business Media: New York, NY, USA, 2011.
13. Andradóttir, S. Simulation optimization. In Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice; John Wiley & Sons: New York, NY, USA, 1998; pp. 307–333.
14. BoussaïD, I.; Lepagnot, J.; Siarry, P. A survey on optimization metaheuristics. Inf. Sci. 2013, 237, 82–117.
15. Glover, F.; Laguna, M.; Marti, R. Scatter search and path relinking: Advances and applications. In Handbook of Metaheuristics; Springer: Boston, MA, USA, 2003; pp. 1–35.
16. Ribeiro, C.; Hansen, P. Essays and Surveys in Metaheuristics; Springer Science & Business Media: New York, NY, USA, 2002.
17. Siarry, P. Metaheuristics; Springer International Publishing: Cham, Switzerland, 2016.
18. Kvasov, D.E.; Mukhametzhanov, M.S. Metaheuristic vs. deterministic global optimization algorithms: The univariate case. Appl. Math. Comput. 2017, 318, 245–259. doi:10.1016/j.amc.2017.05.014.
19. Georgieva, A.; Jordanov, I. Hybrid metaheuristics for global optimization: A comparative study. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer: Berlin/Heidelberg, Germany, 2008, Volume 5271 LNAI, pp. 298–305. doi:10.1007/978-3-540-87656-4_37.
20. Liberti, L.; Kucherenko, S. Comparison of deterministic and stochastic approaches to global optimization. Int. Trans. Oper. Res. 2005, 12, 263–285. doi:10.1111/j.1475-3995.2005.00503.x.
21. Törn, A.; Ali, M.M.; Viitanen, S. Stochastic Global Optimization: Problem Classes and Solution Techniques. J. Glob. Optim. 1999, 14, 437–447. doi:10.1023/A:1008395408187.
22. Serani, A.; Diez, M. Are Random Coefficients Needed in Particle Swarm Optimization for Simulation-Based Ship Design? In Proceedings of the VII International Conference on Computational Methods in Marine Engineering, Nantes, France, May 15–17, 2017.
23. Battiti, R.; Brunato, M.; Mascia, F. Reactive Search and Intelligent Optimization; Springer Science & Business Media: New York, NY, USA, 2008; Volume 45.
24. Jordanov, I.; Georgieva, A. Neural network learning With global heuristic search. *IEEE Trans. Neural Netw.* 2007, 18, 937–942. doi:10.1109/TNN.2007.891633.

25. Lu, Y.; Zhou, Y.; Wu, X. A Hybrid Lightning Search Algorithm-Simplex Method for Global Optimization. *Discrete Dyn. Nature Soc.* 2017, 2017. doi:10.1155/2017/8342694.

26. Blum, C.; Roli, A., Hybrid Metaheuristics: An Introduction. In *Hybrid Metaheuristics: An Emerging Approach to Optimization*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 1–30. doi:10.1007/978-3-540-78295-7_1.

27. D’Andreaiovanni, F.; Krolikowski, J.; Pulaj, J. A fast hybrid primal heuristic for multiband robust capacitated network design with multiple time periods. *Appl. Soft Comput. J.* 2015, 26, 497–507. doi:10.1016/j.asoc.2014.10.016.

28. Renders, J.M.; Flasse, S.P. Hybrid methods using genetic algorithms for global optimization. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* 1996, 26, 243–258. doi:10.1109/3477.485836.

29. Homem-De-Mello, T. Variable-sample methods for stochastic optimization. *ACM Trans. Model. Comput. Simul. (TOMACS)* 2003, 13, 108–133.

30. Hedar, A.; Fukushima, M. Evolution strategies learned with automatic termination criteria. In Proceedings of the SCIS-ISIS, Tokyo, Japan, September 20–24, 2006; pp. 1126–1134.

31. Sulaiman, M.; Salhi, A. A seed-based plant propagation algorithm: The feeding station model. *Sci. World J.* 2015, 2015. doi:10.1155/2015/904364.

32. Laguna, M.; Martí, R. Experimental testing of advanced scatter search designs for global optimization of multimodal functions. *J. Glob. Optim.* 2005, 33, 235–255.

33. Duarte, A.; Martí, R.; Glover, F.; Gortazar, F. Hybrid scatter tabu search for unconstrained global optimization. *Ann. Oper. Res.* 2011, 183, 95–123.

34. Floudas, C.; Pardalos, P.; Adjiman, C.; Esposito, W.; Gumus, Z.; Harding, S.; Klepeis, J.; Meyer, C.; Schweiger, C. *Handbook of Test Problems in Local and Global Optimization*; Springer Science & Business Media: New York, NY, USA, 1999.

35. Hedar, A.; Fukushima, M. Tabu search directed by direct search methods for nonlinear global optimization. *Eur. J. Oper. Res.* 2006, 170, 329–349.

36. He, D.; Lee, L.H.; Chen, C.H.; Fu, M.C.; Wasserkruš, S. Simulation optimization using the cross-entropy method with optimal computing budget allocation. *ACM Trans. Model. Comput. Simul. (TOMACS)* 2010, 20, 4.

37. Michalewicz, Z.; Nazhiyath, G. Genocop III: A co-evolutionary algorithm for numerical optimization problems with nonlinear constraints. In *Proceedings of the 1995 IEEE International Conference on Evolutionary Computation*, Perth, WA, Australia, 29 November–1 December 1995; Volume 2, pp. 647–651.
48. Serani, A.; Leotardi, C.; Iemma, U.; Campana, E.F.; Fasano, G.; Diez, M. Parameter selection in synchronous and asynchronous deterministic particle swarm optimization for ship hydrodynamics problems. *Appl. Soft Comput.* 2016, *49*, 313–334.

49. 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*, 959–977.

50. Sheskin, D.J. *Handbook of Parametric and Nonparametric Statistical Procedures*; CRC Press: Boca Raton, FL, USA; London, UK; New York, NY, USA; Washington, D.C., WA, USA, 2003.

51. Zar, J.H. *Biostatistical Analysis*; Pearson New International Edition: Essex, England, Fifth Edition, 2013.

52. 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.

53. García, S.; Molina, D.; Lozano, M.; Herrera, F. A study on the use of non-parametric tests for analyzing the evolutionary algorithms’ behaviour: A case study on the CEC’2005 special session on real parameter optimization. *J. Heuristics* 2009, *15*, 617–644.

54. Hansen, N. The CMA evolution strategy: a comparing review. In *Towards a New Evolutionary Computation*; Springer-Verlag: Berlin/Heidelberg, Germany, 2006; pp. 75–102.