A Multilevel Splitting Algorithm for Quick Sampling

Liping Wang* and Wenhui Fan

Abstract: To reduce intermediate levels of splitting process and enhance sampling accuracy, a multilevel splitting algorithm for quick sampling is proposed in this paper. Firstly, the selected area of the elite set is expanded to maintain the diversity of the samples. Secondly, the combined use of an adaptive difference evolution algorithm and a local searching algorithm is proposed for the splitting procedure. Finally, a suite of benchmark functions are used for performance testing. The results indicate that the convergence rate and stability of this algorithm are superior to those of the classical importance splitting algorithm and an adaptive multilevel splitting algorithm.

Key words: rare event simulation; splitting method; elite set; adaptive differential evolution; local searching

1 Introduction

As is well known, splitting\(^{[1]}\) is a classical algorithm for rare event simulation. Its objective is to break down a definitional domain into a train of nested subsets to search for rare samples of excellent individuals, and to evaluate their probability by calculating the percentage of success for a total samples. To improve flexibility and efficiency, many splitting algorithms have been developed in succession. Botev\(^{[2]}\) proposed an adaptive multilevel splitting algorithm, often simply referred to as Adaptive Multilevel splitting (ADAM) for choosing the splitting level in an adaptive manner and obtaining candidates by the artificial construction of Markov chains. Wadman et al.\(^{[3]}\) proposed an accelerating splitting algorithm and studied the rate functions to optimize the path. Louvin et al.\(^{[4]}\) presented an adaptive multilevel splitting algorithm for application to particle transport.

In a previous work, we proposed a multilevel Splitting algorithm based on Difference Evolution (SDE)\(^{[5]}\), which embeds the technique of Difference Evolution (DE) into the splitting framework for generating candidates. In this paper, we propose an improved splitting algorithm to reduce the intermediate levels of the splitting process. For simplicity, we refer to this algorithm as the SADE-L algorithm.

The rest of this paper is organized as follows. We briefly introduce the proposed multilevel splitting algorithm in Section 2 and provide details of the SADE-L algorithm in Section 3. The performances of SADE-L, classical Importance Splitting (ISp), and the ADAM algorithms are compared with respect to a set of more challenging benchmark functions in Section 4. We draw our conclusion in Section 5.

2 Background

For a real-value function \(f(X), X \in \mathcal{X}\), we are interested in estimating the probability that it is less than a fixed threshold \(\gamma\): \(p = \mathbb{P}(f(X) \leq \gamma)\), where \(X = (x_1, x_2, \ldots, x_D)\) is a \(D\)-dimension random variable and subject to the probability distribution \(g(X)\) on \(\mathcal{X}\), such that when \(\gamma\) decreases, \(p\) goes to zero, \(\{f(X) \leq \gamma\}\) becomes a rare event, and \(\mathbb{P}(f(X) \leq \gamma)\) is the occurrence probability of the rate event. The basic idea of the multilevel splitting algorithm is to transform rare events into a series of nested non-small probability subsets \(X_k = \{f(X) \leq \gamma_k\}\) by the intermediate levels \(\gamma_1 < \cdots < \gamma_k < \cdots < \gamma_m\). Then, the evaluated rare event probability is converted into the product of many conditional probabilities, \(\hat{p} = \hat{p}_1 \hat{p}_2 \cdots \hat{p}_m\). Furthermore, the conditional probabilities

\*Liping Wang and Wenhui Fan are with the Department of Automation, Tsinghua University, Beijing 100084, China. E-mail: wanglp14@mails.tsinghua.edu.cn; fanwenhui@tsinghua.edu.cn.

\*To whom correspondence should be addressed.

Manuscript received: 2019-09-29; revised: 2020-01-14; accepted: 2020-02-09

© The author(s) 2021. The articles published in this open access journal are distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/).
can be computed by the proportion of individuals that are successful in reaching level $k$ in $N_k$ samples, that is, 
\[ \hat{p}_k = \sum_{i=1}^{N_k} f^{(k)}(X_i^{(k)}) / N_k, k = (1, \ldots, m). \]
There are different interpretations regarding the choice of intermediate levels $\gamma_k$ and the splitting strategy.

### 2.1 ISp algorithm

- **Numbered lists.** Determine the intermediate levels $\gamma_k, k = 1, \ldots, m$. Define $g_k$ as the density of the set $X \in X_k$ and $\mu_k$ as the density of the corresponding $f(X)$. $X_1^{(k)}, \ldots, X_N^{(k)}$ are samples generated from $g_k$ and the estimator of the $\alpha$-quantile of $\mu_k$ is $Q^{(k)}_\alpha$. Let $\gamma_{k+1} = Q^{(k)}_\alpha$ and $X_{k+1} = \{f(X) < \gamma_{k+1} = Q^{(k)}_\alpha\}$.

- **Splitting.** Apply the Metropolis Hastings algorithm to simulate a Markovian reversible kernel for a density $X$ and use the proposal/refusal algorithm to generate candidates. Repeat the steps above and estimate the probability $\hat{p}_{\text{ISp}} = (1 - \alpha)^k \times \sum_{i=1}^{N_k} f(X_i^{(k)}) < \gamma / N_k$. More details can be seen in Ref. [6].

### 2.2 ADAM algorithm

- **Numbered lists.** Determine the intermediate splitting levels $\gamma_k, k = 0, \ldots, m$. Compute the values of $f(X)$ for each $X \in X_k$, $k = 0, \ldots, m$, and sort them in ascending order $f_1 \leq f_2 \leq \cdots \leq f_N, N$ is the sample size of $X_k$. Set $\gamma_{k+1} = f_{N_{\rho}}, \ N_{\rho} = N \times \rho$ and $\rho \in (0, 1]$ is a rare parameter. The elements $\{X \in X_k : f(X) \leq \gamma_{k+1}\}$ are preserved as the elite set $E_{k+1}$ for splitting in the next level.

- **Splitting.** ADAM artificially generates offspring in the form of a Markov chain for each starting sample artificially [7,8]. Two schemes are used to qualify the number of offspring. The first is fixed splitting whereby each elite point has the same split factor $s$. The other one is fixed effort, which we use in this paper, whereby there are equal numbers of total candidates in each level. In this scheme, for each $X \in E_{k+1}$, the splitting factor $s_f$ is defined as follows [9]:

\[
s_f = \left[ \frac{N}{N_{E}} \right] + b_i, i = 1, \ldots, N_E \tag{1}
\]

where $b_1, \ldots, b_{N_{E}}$ are random variables subject to Bernoulli distributions that satisfy $\sum b_i = N \mod N_E$. Repeat the above steps and estimate the probability with $\hat{p}_{\text{ADAM}} = \prod_{k=1}^{m} \rho_k = \prod_{k=1}^{m} \frac{R_k}{N}, R_k = \sum_{i=1}^{N} f(X_i^{(k)}) \leq \gamma_k$.

As shown in Fig. 1, within the framework of a multilevel splitting algorithm in 3-D space, first, randomly create an initial sample set $X_0 = \{X_1, X_2, X_3, X_4, X_5\}$ and suppose the rare parameter $\rho = 0.8$. Then, choose the threshold $\gamma_1 = f_5$, and the individual $X_1$ and $X_3$ then save in the elite set $E_1$ for constructing population $X_1$ in the second level. Five candidates are split in the Fixed Effort (FE) scheme, two of which are put into elite set $E_2$ for splitting in the next level. Repeat the above steps until there are points in the rare event set.

The SADE-L algorithm applies the same splitting framework as ADAM, but a different construction of the elite set and splitting mechanism are proposed, as detailed in Section 3.

### 3 SADE-L Algorithm

The ISp and ADAM algorithms can resolve the rare event problem well. However, with increased complexity, the number of intermediate level increases greatly, which slows the process. To address this problem, in this paper, we propose the SADE-L algorithm, which extends the selection range of the elite set to improve the diversity of the start states. The splitting process is executed using an adaptive differential evolution algorithm and a local searching algorithm, which reduces the number of intermediate levels. The details of this process are shown below.

First, randomly generate a population $X_{k} = (X_{1}^{(k)}, \ldots, X_{N}^{(k)})$, where $k$ denotes the splitting level with a start value of zero and $X_{1}^{(k)} = (x_{1}^{(k)}, \ldots, x_{D}^{(k)})$, and $i=(1, \ldots, N)$ is the $D$-dimension decision variance, with the upper bound vector $U = (u_1, \ldots, u_D)$ and the lower bound vector $L=(l_1, \ldots, l_D)$. The corresponding

![Fig. 1 Schematic diagram of multilevel splitting algorithm in 3-D space.](image-url)
objective function is \( F^{(k)} = (f_1^{(k)}, \ldots, f_N^{(k)}) \). Then determine whether the terminal condition is met. The process ends when the probability is less than a specified value or the splitting level exceeds the maximum level. Otherwise, the process continues to the next step.

Next, evaluate the objective function values \( F^{(k)} \) and sort them in increasing order. From the samples collected from the current and previous levels, choose the \( N_E \) individuals ranked at the front for membership in the elite set \( E_k \). Details are provided in Section 3.1.

Thirdly, execute the splitting progress using the FE scheme by combining adaptive differential evolution with local searching, and update the decision vector \( X_i^{(k)} \), \( i = 1, \ldots, N \), on each dimension.

We use an alternative approach to randomly assign the sampling method for a vector \( X_i^{(k)}(d) \). In the adaptive DE algorithm, we allocate a mutation factor \( F = (F_1, \ldots, F_D) \) and a cross-over factor \( CR = (CR_1, \ldots, CR_D) \) for each dimension. Initially, \( F^0 = (0.5, \ldots, 0.5) \) and \( CR^0 = (0.8, \ldots, 0.8) \). If the intermediate point is inferior to the start point, then update \( F(d) \) and \( CR(d) \), \( d = 1, 2, \ldots, D \), using Eqs. (6) and (9), respectively. In the local searching procedure, we impose a disturbance on the start point in the positive or negative direction to search for the points that are spatially closer to the rare set. More details are discussed in Section 3.2. As the splitting process progress, a rare set is ultimately obtained. Figure 2 shows a flow diagram of the SADE-L algorithm.

The pseudo-code of the SADE-L algorithm is presented in Algorithm 1.

### Algorithm 1 Pseudo-code of the SADE-L algorithm

**Input:** Population size \( N \), rare parameter \( \rho \), upper bound \( U \), lower bound \( L \) of the decision vectors, initial mutation factor \( F^0 \), and initial crossover factor \( CR^0 \)

**Output:** Rare event probability \( p \) and the relative error RE

1. Generate initial population \( X_0 = (X_1^{(0)}, \ldots, X_N^{(0)}) \) and calculate the objective functions \( F^{(0)} = (f^{(0)}(X_1), \ldots, f^{(0)}(X_N)) \)
2. for \( t = 1: \) iteration do
3. While the stop criteria is not matched do
4. Construct the elite set: establish elite set \( E_{k+1} \) according to Algorithm 2 and calculate the splitting factor \( sf \) for each element according to Eq. (1)
5. for \( i = 1: N_E \) do
6. for \( j = 1: sf \) do
7. Choose \( r_1, r_2, \) and \( r_3 \) from \( \{1, \ldots, N_E\}/i \)
8. for \( i = 1: D \) do
9. if rand<0.5 then
10. Generate offspring according to Algorithm 3
11. else
12. Generate offspring according to Algorithm 4
13. end if
14. end for
15. end for
16. end for
17. Calculate the rare event probability \( p(t) \)
18. end while
19. \( t = t + 1 \)
20. end for
21. Return the mean probability \( p \) and the relative error RE

![Flow diagram of the SADE-L algorithm.](image-url)
Algorithm 2 Selection of the elite set in each splitting level

```
Input: Samples of \( X_0 = \{ X^{(0)}_1, \ldots, X^{(0)}_N \} \) and the objective vectors \( F_0 = \{ f^{(0)}_1, \ldots, f^{(0)}_N \} \)
Output: Elite set \( E_k \) and the threshold \( y_k 
1: \text{if } k < 1 \text{ then}
2: \quad \text{for the samples } X_0', X_0 = \{ X^{(0)}_1, \ldots, X^{(0)}_N \}, \text{calculate the objective vectors } F_0 \text{ and sort it from the smallest to the largest as } f^{(0)}_1 < f^{(0)}_2 < \ldots < f^{(0)}_N \}
3: \quad \text{end if}
4: \quad \text{if } X^{(0)}_i \neq X^{(0)}_j \quad \text{for the target component } \quad \text{end if}
5: \quad \text{if } X^{(0)}_i \neq X^{(0)}_j \quad \text{end if}
6: \quad \text{end if}
7: \quad \text{end if}
8: \quad \text{end if}
9: \quad \text{end if}
10: \quad \text{end if}
```

Algorithm 3 Adaptive DE algorithm for splitting

```
Input: Target point \( X_i^{(k)} = (X_i^{(k)}(1), \ldots, X_i^{(k)}(D)) \)
Output: Updated \( X_i^{(k+1)} = (X_i^{(k+1)}(1), \ldots, X_i^{(k+1)}(D)) \)
1: Mutation: select three different individuals \( r_1, r_2, \) and \( r_3 \) randomly
2: for the target component \( X_i^{(k)}(d) \), construct \( X_i^{(k)}(d) = X_i^{(k)}(d) + F(d) \cdot (X_{r_2}^{(k)}(d) - X_{r_3}^{(k)}(d)) \)
3: if \( X_i^{(k)}(d) < L(d) \) or \( X_i^{(k)}(d) > U(d) \) then
4: Reset \( X_i^{(k)}(d) \) according to Eq. (5)
5: end if
6: Update \( F(d) \) according to Eq. (6)
7: if \( F(d) < 0 \) or \( F(d) > 1 \) then
8: Reset \( F(d) = 0.5 \)
9: end if
10: Cross-over: Construct \( X_i^{(k+1)}(d) = X_i^{(k)}(d) \)
11: Update CR(d) according to Eq. (9)
12: if CR(d) < 0.5 or CR(d) > 1 then
13: Reset CR(d) = 0.8
14: end if
15: Selection: compare the dominance between \( f(X_i^{(k)}) \) and \( f(X_i^{(k)}) \)
16: if \( f(X_i^{(k)}) \leq f(X_i^{(k)}) \) then
17: \( X_i^{(k+1)} = X_i^{(k)}(d) \)
18: else \( f(X_i^{(k)}) > f(X_i^{(k)}) \) then
19: \( X_i^{(k+1)} = X_i^{(k)}(d) \)
20: end if
```

the objective function value \( F^{(0)} = \{ f^{(0)}_1, \ldots, f^{(0)}_N \} \) of each sample of \( X_0' \) as \( f^{(0)}_1 < \ldots < f^{(0)}_N \), and choose \( E_k = \{ X^{(0)}_1, \ldots, X^{(0)}_N \} \).

- If \( k \geq 1 \), let \( X''_k = \{ X_k \cup E_k = \{ X^{(k)}_1, \ldots, X^{(k)}_N \}, \ldots, X^{(k-1)} \} \). The sorted function is \( f^{(k)}_1 < \ldots < f^{(k)}_N \), and use the best \( N_E \) samples to construct the elite set \( E_k \), where \( J, i = 1, \ldots, N_E \).

Algorithm 4 Local searching algorithm for splitting

```
Input: Target point \( X_i^{(k)} = (X_i^{(k)}(1), \ldots, X_i^{(k)}(D)) \), \( f(X_i^{(k)}) \), and the best value \( x_i^{(k)} \) and the worst value \( x_i^{(k)} \) of \( d \)-th dimension in the \( k \)-th level
Output: Updated \( X_i^{(k+1)} = (X_i^{(k+1)}(1), \ldots, X_i^{(k+1)}(D)) \), \( f(X_i^{(k+1)}) \)
1: Searching: for each component \( X_i^{(k)}(d) \), construct \( X_i^{(k)}(d) \) according to Eq. (11)
2: if \( X_i^{(k)}(d) < L(d) \) or \( X_i^{(k)}(d) > U(d) \) then
3: Reset \( X_i^{(k)}(d) \) according to Eq. (5)
4: end if
5: Selection: compare the value between \( f(X_i^{(k)}) \) and \( f(X_i^{(k)}) \)
6: if \( f(X_i^{(k)}) \leq f(X_i^{(k)}) \) then
7: \( X_i^{(k+1)}(d) = X_i^{(k)}(d) \)
8: else \( f(X_i^{(k)}) > f(X_i^{(k)}) \) then
9: \( X_i^{(k+1)}(d) = X_i^{(k)}(d) \)
10: end if
```

comprises the subscript index of the sorted order.

### 3.2 Splitting mechanism of the SADE-L algorithm

In previous work, we integrated the DE algorithm into the splitting framework to produce the next generation and obtained a good global optimization result. On this basis, here, we propose a novel splitting mechanism that combines an adaptive differential evolution algorithm with a local searching algorithm. We update the intermediate point using a dimension-by-dimension approach to improve the diversity.

#### (1) Adaptive differential evolution algorithm

Define the elite population as

\[
\begin{pmatrix}
X_1^{(k)}(1) & \cdots & X_1^{(k)}(D) \\
\vdots & \ddots & \vdots \\
X_{N_E}^{(k)}(1) & \cdots & X_{N_E}^{(k)}(D)
\end{pmatrix}
\]  

The main steps of the adaptive DE algorithm in each iteration are as follows:

- **Mutation.** For the \( i \)-th target point \( X_i^{(k)} = (X_i^{(k)}(1), \ldots, X_i^{(k)}(D)) \), \( i = 1, \ldots, N_E \), \( N_E \) is the size of the elite set and \( k \) denotes the splitting level. The mutation point \( X_i^{(k)} = (X_i^{(k)}(1), \ldots, X_i^{(k)}(D)) \) of the DE/rand/1/bin strategy is:

\[
X_i^{(k)} = X_i^{(r_1)} + F \cdot (X_{r_2}^{(k)} - X_{r_3}^{(k)})
\]

where \( r_1, r_2, \) and \( r_3 \) are three unequal random and uniform individuals of set \{1, 2, 3, \ldots, \( N_E \)\}/i. \( F \) is a magnification factor belonging to \([0, 2]\).

To enhance the validity of the offspring population, we propose the use of a novel adaptive DE algorithm for adaptively choosing the control parameter \( F \). We assign \( F = [0.5, 0.5] \times 2 \times D \) to start. Then, Eq. (3) can be
rewritten as follows:
\[ \hat{X}_i^{(k)}(d) = X_i^{(k)}(d) + \frac{F(d) - (X_i^{(k)}(d) - X_j^{(k)}(d))}{\sigma_i^2(d)} \] (4)
where \( d = 1, 2, \ldots, D \). If \( \hat{X}_i^{(k)}(d) \) exceeds the range of the decision variance, reset the mutation point as follows:
\[ \hat{X}_i^{(k)}(d) = \begin{cases} L(d), & \text{if } \hat{X}_i^{(k)}(d) < L(d) \\ U(d), & \text{if } \hat{X}_i^{(k)}(d) > U(d) \end{cases} \] (5)
where \( U(d) \) and \( L(d) \) are the upper and lower bounds of the \( d \)-th dimension, respectively.

In this process, if the mutation point \( \hat{X}_i^{(k)}(d) \) is worse than the target point \( X_i^{(k)}(d) \), choose \( F(d) \) in the same dimension of the next generation as shown in Eq. (6). Reset \( F(d) = 0.5 \) when it is less than 0 or greater than 1. Otherwise, reserve \( F(d) \) for the next generation, that is,
\[ F(d) = \begin{cases} F(d) + 0.1 \times N(0, 1), & \text{if } f(X_i^{(k)}) < f(\hat{X}_i^{(k)}) \\ F(d), & \text{otherwise} \end{cases} \] (6)

- **Cross-over**. Diversity can be improved with a cross-over factor. For simplicity, this is set to be a constant in the range of \([0, 1]\) (e.g., \( CR = 0.8 \)) to determine whether the trail point inherits the value of the mutation point or remains unchanged, that is,
\[ \tilde{X}_i^{(k)} = \begin{cases} \hat{X}_i^{(k)}, & \text{if rand}(0, 1) \leq CR; \\ X_i^{(k)}, & \text{otherwise} \end{cases} \] (7)

In SADE-L, \( CR = [0.8, \ldots, 0.8]_{N \times D} \) at the start, and Eq. (7) can be rewritten as follows:
\[ \tilde{X}_i^{(k)}(d) = \begin{cases} \hat{X}_i^{(k)}(d), & \text{if rand}(0, 1) \leq CR(d); \\ X_i^{(k)}(d), & \text{otherwise} \end{cases} \] (8)

Generate \( CR(d) \) as shown in Eq. (9) if the candidate fails and reset it to equal 0.8 when it goes beyond the boundaries. Otherwise, reserve \( CR(d) \) for the next generation.
\[ CR(d) = \begin{cases} CR(d) + 0.1 \times N(0, 1), & \text{if } f(\tilde{X}_i^{(k)}) < f(X_i^{(k)}) \\ CR(d), & \text{otherwise} \end{cases} \] (9)

- **Selection**. Compute the objective function values of \( X_i^{(k)}(d) \) and \( \tilde{X}_i^{(k)}(d) \), then make a comparison to choose a solution that can survive to the next level.
\[ X_i^{(k+1)}(d) = \begin{cases} \tilde{X}_i^{(k)}(d), & \text{if } f(\tilde{X}_i^{(k)}) < f(X_i^{(k)}) \\ X_i^{(k)}(d), & \text{otherwise} \end{cases} \] (10)

(2) **Local searching algorithm**

For global searching, we use an adaptive difference evolution algorithm. In addition, a local searching algorithm is utilized to obtain good solutions in a neighborhood. In this step, the intermediate is generated based on a local displacement of the start point in each dimension, and the vector with a superior objective function value is retained for the next level.

- **Searching**. For the \( i \)-th target point \( X_i^{(k)} = (X_i^{(k)}(1), \ldots, X_i^{(k)}(D)) \), the candidate point \( \hat{X}_i^{(k)} = (\hat{X}_i^{(k)}(1), \ldots, \hat{X}_i^{(k)}(D)) \) can be calculated as follows:
\[ \hat{X}_i^{(k)}(d) = X_i^{(k)}(d) + \beta \cdot (X_B^{(k)}(d) - X_W^{(k)}(d)) \] (11)
where \( \beta \sim U(0, 1) \) is a reduction factor. \( X_B^{(k)}(d) \) and \( X_W^{(k)}(d) \) denote the decision vectors of the best and worst objective functions that correspond to the \( d \)-th dimension in the \( k \)-th level, respectively.

- **Selection**. Compare the relationship between vectors \( f(X_i^{(k)}) \) and \( f(\hat{X}_i^{(k)}) \) and select the element of \( X_{k+1} \) shown in Eq. (12):
\[ X_{k+1}(d) = \begin{cases} \hat{X}_i^{(k)}(d), & \text{if } f(\hat{X}_i^{(k)}) \leq f(X_i^{(k)}) \\ X_i^{(k)}(d), & \text{otherwise} \end{cases} \] (12)

4 **Experiment Setting and Comparison**

Next, we evaluate the SADE-L algorithm against 24 complex benchmark functions of the CEC’05 test set\(^{10}\), which are often used for testing the global optimal algorithm. Suo et al.\(^{9}\) studied the equivalence relation between an extreme problem and a rare event, that is, the extremum problem can be considered to be a special case of a rare event under certain conditions. Morio et al.\(^{6}\) also applied an optimal function to evaluate the performance of a rare event simulation algorithm, due to the significant difficulty of multidimensional sampling. These studies provide a theoretical basis for experimentation. Only the minimization problem is considered in this paper. The results are used to compare the performances of SADE-L, ADAM, and ISp algorithms.

The SADE-L algorithm was coded in MATLAB and executed on a computer with Intel(R) Core(TM) i7-5500U CPU @2.40 GHz. The results were run 10 times independently to compute the mean rare event probability \( p \), the number of intermediate levels \( T \), and the relative error \( RE \) for the given threshold \( \gamma \).

The performance evaluations comprised two parts. In Part 1, we used the 30-dimension functions to compare the performances of the SADE-L, ADAM, and ISp algorithms. In Part 2, we increased the dimensions of the test functions to 50, which is much more difficult to handle. Table 1 shows the specific parameter settings we used and Table 2 lists the test functions.

(1) **Part 1**

The values of \( \gamma \) must be allocated prior to beginning
Table 1 Parameter settings of the three algorithms.

| Part | N    | D   | \(\rho\)_{SADE-L} | \(\rho\)_{ADAM} | \(\rho\)_{ISp} |
|------|------|-----|-------------------|-----------------|----------------|
| Part 1 | 1000 | 30  | 0.5               | 0.5             | 0.5            |
| Part 2 | 1500 | 50  | 0.5               | 0.5             | 0.5            |

Table 2 Functions used in performance testing.

| \(f\) | Function                                      |
|-------|-----------------------------------------------|
| \(f_1\) | Shifted Sphere Function                       |
| \(f_2\) | Shifted Schwefel’s Problem 1.2               |
| \(f_3\) | Shifted and Rotated High Conditioned Elliptic Function |
| \(f_4\) | Shifted Schwefel’s Problem 1.2 with Noise in Fitness |
| \(f_5\) | Schwefel’s Problem 2.6 with Global Optimum on Bounds |
| \(f_6\) | Shifted Rosenbrock’s Function                 |
| \(f_7\) | Shifted Rotated Griewank’s Function without Bounds |
| \(f_8\) | Shifted and Rotated Ackley’s Function with Global Optimum on Bounds |
| \(f_9\) | Shifted Rastrigin’s Function                  |
| \(f_{10}\) | Shifted and Rotated Rastrigin’s Function      |
| \(f_{11}\) | Shifted Rotated Weierstrass Function          |
| \(f_{12}\) | Schwefel’s Problem 2.13                       |
| \(f_{13}\) | Expanded Extended Griewank’s plus Rosenbrock’s Function |
| \(f_{14}\) | Expanded Extended Scafe’s F6                  |
| \(f_{15}\) | Hybrid Composition Function 1                |
| \(f_{16}\) | Rotated Hybrid Composition Function 1         |
| \(f_{17}\) | Rotated Hybrid Composition Function 1 with Noise in Fitness |
| \(f_{18}\) | Rotated Hybrid Composition Function 2         |
| \(f_{19}\) | Rotated Hybrid Composition Function 2 with a Narrow Basin for the Global Optimum |
| \(f_{20}\) | Rotated Hybrid Composition Function 2 with the Global Optimal on the Bounds |
| \(f_{21}\) | Rotated Hybrid Composition Function 3         |
| \(f_{22}\) | Rotated Hybrid Composition Function 3 with High Condition Number Matrix |
| \(f_{23}\) | Non-Continuous Rotated Hybrid Composition Function 3 |
| \(f_{24}\) | Rotated Hybrid Composition Function 4         |

the experiments. In Part 1, we set the value of \(\gamma\) to make the rare event probability \(\hat{p} < 10^{-9}\) or the approximate minimum value obtainable within the maximization splitting level \(L = 50\) for the ISp algorithm.

In this part, the population size is \(N = 1000\) and the dimension is \(D = 30\). Table 3 shows the numerical result, where \(A_1\), \(A_2\), and \(A_3\) represent the SADE-L, ADAM, and ISp algorithms, respectively.

In Table 3, we can see the SADE-L algorithm has fewer intermediate levels than the ADAM and ISp algorithms for all 24 functions at a given threshold, which is shown in bold. For the function \(f_{18}\), \(f_{19}\), and \(f_{20}\), since the ADAM algorithm cannot obtain a rare event set within 50 intermediate levels for \(D = 30\), the results are represented by “\(x\)”. The ISp algorithm also required more than 30 splitting processes to obtain a rare event set, whereas, the SADE-L algorithm needed only eight intermediate levels. For the function \(f_3\), \(f_5\), and \(f_7\), the intermediate levels of ISp are four times those of SADE-L, respectively; for function \(f_16\), \(f_{17}\), \(f_{22}\), and \(f_{24}\), the intermediate levels of ISp are four times those of SADE-L. For functions \(f_3\), \(f_5\), \(f_6\), \(f_{15}\), \(f_{16}\), and \(f_{24}\), the intermediate levels of ADAM are double those of SADE-L.

In terms of RE, SADE-L has the minimum value in functions \(f_2\), \(f_3\), \(f_4\), \(f_5\), \(f_6\), \(f_7\), \(f_8\), \(f_{12}\), \(f_{13}\), \(f_{17}\), \(f_{18}\), \(f_{19}\), \(f_{20}\), and \(f_{24}\) for \(D = 30\).

Figure 3 shows the thresholds of function \(f_{10}\) at 30-dimension after the \(k\)-th splitting to enable a comparison of the convergence rate of the three algorithms. In the minimization problem, the rare event set will be realized as the threshold decreases to \(\gamma\). It is easy to see that the algorithm with the minimum threshold is the best in terms of convergence. For a given threshold, SADE-L also reaches this threshold with the fewest splitting levels.

(2) Part 2

In Part 2, we set the value of \(\gamma\) to make the rare event probability \(\hat{p} < 10^{-9}\) or the approximate minimum value obtainable within the maximization splitting level \(L = 100\) for the SADE-L algorithm. In this part, the population size was \(N = 1500\) and the dimension was \(D = 50\). Since the ISp algorithm is unavailable for most of the functions in 50-dimension, it is ignored in this part and we compare only the performances of SADE-L and ADAM. Table 4 shows the numerical results.

In Table 4, there are fewer intermediate levels \(T\) for the SADE-L algorithm than ADAM for all the 24 functions at the given threshold, which is shown in bold. For example, the ADAM algorithm could not obtain a rare event set in functions \(f_1\), \(f_4\), \(f_5\), \(f_7\), \(f_{17}\), \(f_{18}\), \(f_{19}\), and \(f_{20}\) within 100 intermediate levels at \(D = 50\), so the results are represented by “\(x\)”. In contrast, the SADE-L algorithm obtained rare event samples after approximately 30 splitting processes. In addition, the SADE-L algorithm demonstrated a clear advantage in functions \(f_9\), \(f_{10}\), \(f_{13}\), \(f_{16}\), and \(f_{24}\). Thus, SADE-L needed fewer levels than ADAM in this process, which proves that SADE-L exhibited greater efficiency.
Table 3  Means of the rare event probability $\hat{p}$ over 10 runs, the number of intermediate levels $T$, and the relative error RE of the three algorithms, with $D = 30$ and $N = 1000$.

|   | $f$       | $\gamma$   | $\hat{p}$  | $T$   | RE (%) |
|---|-----------|-------------|-------------|-------|--------|
|   |           |             | $A_1$       | $A_2$ | $A_3$  | $A_1$ | $A_2$ | $A_3$ |
| $f_1$ | 73 900 | $6.80 \times 10^{-3}$ | $2.29 \times 10^{-3}$ | $8.03 \times 10^{-5}$ | 7 | 8 | 13 | 20.68 | 6.51 | 83.21 |
| $f_2$ | $100 000$ | $2.62 \times 10^{-2}$ | $4.27 \times 10^{-3}$ | $7.62 \times 10^{-9}$ | 6 | 8 | 27 | 4.68 | 5.32 | 80.27 |
| $f_3$ | $2.65 \times 10^9$ | $1.19 \times 10^{-1}$ | $1.94 \times 10^{-2}$ | $2.61 \times 10^{-10}$ | 3 | 6 | 32 | 2.81 | 3.34 | 28.97 |
| $f_4$ | 110 000 | $3.60 \times 10^{-3}$ | $3.86 \times 10^{-4}$ | $1.94 \times 10^{-10}$ | 7 | 13 | 32 | 13.60 | 20.82 | 99.13 |
| $f_5$ | 53 500 | $8.60 \times 10^{-1}$ | $2.84 \times 10^{-2}$ | $2.79 \times 10^{-5}$ | 2 | 4 | 14 | 1.07 | 4.02 | 95.37 |
| $f_6$ | $3.78 \times 10^{10}$ | $6.82 \times 10^{-2}$ | $2.64 \times 10^{-5}$ | $4.70 \times 10^{-6}$ | 5 | 13 | 18 | 5.74 | 88.24 | 99.99 |
| $f_7$ | 3860 | $8.07 \times 10^{-1}$ | $6.62 \times 10^{-3}$ | $6.90 \times 10^{-6}$ | 2 | 7 | 17 | 3.33 | 4.92 | 100 |
| $f_8$ | 21.03 | $2.34 \times 10^{-5}$ | $1.30 \times 10^{-6}$ | $2.24 \times 10^{-6}$ | 15 | 18 | 18 | 45.39 | 72.94 | 89.39 |
| $f_9$ | 680 | $2.45 \times 10^{-3}$ | $3.16 \times 10^{-4}$ | $2.96 \times 10^{-6}$ | 8 | 11 | 18 | 24.11 | 9.60 | 79.62 |
| $f_{10}$ | 680 | $2.39 \times 10^{-3}$ | $2.87 \times 10^{-4}$ | $7.36 \times 10^{-6}$ | 8 | 12 | 16 | 27.38 | 5.86 | 49.04 |
| $f_{11}$ | 42 | $1.59 \times 10^{-4}$ | $2.90 \times 10^{-5}$ | $3.05 \times 10^{-5}$ | 13 | 14 | 15 | 19.64 | 6.70 | 100 |
| $f_{12}$ | $1.4 \times 10^6$ | $1.20 \times 10^{-3}$ | $1.72 \times 10^{-4}$ | $1.36 \times 10^{-5}$ | 8 | 12 | 16 | 13.05 | 14.57 | 85.01 |
| $f_{13}$ | 30 | $1.03 \times 10^{-4}$ | $6.70 \times 10^{-12}$ | $1.09 \times 10^{-6}$ | 12 | 36 | 19 | 4.76 | 7.49 | 93.70 |
| $f_{14}$ | 3.37 | $2.71 \times 10^{-4}$ | $2.54 \times 10^{-6}$ | $7.20 \times 10^{-8}$ | 11 | 18 | 25 | 15.96 | 11.83 | 99.99 |
| $f_{15}$ | 1230 | $7.10 \times 10^{-3}$ | $2.71 \times 10^{-4}$ | $6.14 \times 10^{-7}$ | 6 | 12 | 20 | 18.11 | 8.08 | 99.99 |
| $f_{16}$ | 1250 | $1.28 \times 10^{-2}$ | $5.93 \times 10^{-4}$ | $8.44 \times 10^{-7}$ | 5 | 11 | 20 | 21.30 | 17.37 | 85.92 |
| $f_{17}$ | 1630 | $1.95 \times 10^{-2}$ | $6.87 \times 10^{-3}$ | $3.81 \times 10^{-7}$ | 5 | 6 | 21 | 3.23 | 8.83 | 99.99 |
| $f_{18}$ | 900.20 | $1.90 \times 10^{-3}$ | × | $3.78 \times 10^{-11}$ | 8 | × | 34 | 25.36 | × | 27.42 |
| $f_{19}$ | 900.30 | $1.55 \times 10^{-3}$ | × | $3.37 \times 10^{-10}$ | 8 | × | 31 | 15.54 | × | 54.52 |
| $f_{20}$ | 900.40 | $3.86 \times 10^{-3}$ | × | $9.76 \times 10^{-10}$ | 8 | × | 30 | 1.17 | × | 35.88 |
| $f_{21}$ | 1381 | $5.68 \times 10^{-4}$ | $6.86 \times 10^{-5}$ | $7.35 \times 10^{-6}$ | 11 | 13 | 16 | 11.86 | 7.32 | 73.00 |
| $f_{22}$ | 1800 | $1.16 \times 10^{-2}$ | $1.43 \times 10^{-3}$ | $7.65 \times 10^{-10}$ | 6 | 7 | 29 | 15.48 | 4.87 | 99.52 |
| $f_{23}$ | 1370 | $7.75 \times 10^{-4}$ | $2.01 \times 10^{-5}$ | $3.42 \times 10^{-7}$ | 10 | 14 | 21 | 25.19 | 5.98 | 100 |
| $f_{24}$ | 1450 | $1.20 \times 10^{-3}$ | $1.39 \times 10^{-5}$ | $4.05 \times 10^{-11}$ | 8 | 15 | 33 | 12.85 | 14.22 | 99.91 |

Fig. 3  Evolutions of SADE-L, ADAM, and ISp algorithms for function $f_{10}$ in 30-dimension.

in generating offspring and a higher convergence rate than ADAM at $D = 50$.

In terms of RE, SADE-L achieved the minimum value in most of the 24 functions for $D = 50$.

Figure 4 shows the evolution process of function $f_{10}$ at 50-dimension after the $k$-th splitting. Figure 4 also shows that the convergence rate of SADE-L is fastest and that the fewest levels are required to obtain a rare set minimum. The process of ISp tends to level off gradually with increases in the splitting.

5 Conclusion

In this paper, we proposed a novel multilevel splitting algorithm. Firstly, the rule for creating an elite set was reconstructed, and the selection range was expended to reduce the degree of overlap of elite individuals in adjacent levels. Secondly, we proposed a novel sampling method that combines an adaptive DE algorithm with a local search algorithm. The purpose of this method is to increase the number of the candidates acquired. In the experiments, we used 24 complex benchmark functions to test the features of the SADE-L algorithm. The results indicated that SADE-L demonstrates superior performance in both reducing the number of intermediate levels and increasing stability, as compared to the ADAM and ISp algorithms, which mean it provides an effective arithmetic technique for rare event simulation.

In the future work, we will focus on widening the application of the SADE-L algorithm in global optimization and analyze its convergence for achieving
Table 4 Means of the rare event probability $\hat{\rho}$ over 10 runs, the number of intermediate levels $T$, and the relative errors RE of the three algorithms, with $D = 50$ and $N = 1500$.

| f    | $\gamma$ | $\hat{\rho}$ | $T$ | RE (%) |
|------|-----------|---------------|-----|--------|
| $f_1$ | 75000     | $8.30 \times 10^{-10}$ | 30 | ×      | 30.49 | ×      |
| $f_2$ | 80000     | $2.08 \times 10^{-10}$ | 32 | 35     | 10.21 | 11.51 |
| $f_3$ | $4.6 \times 10^9$ | $2.57 \times 10^{-10}$ | 32 | 34     | 11.35 | 54.38 |
| $f_4$ | 10000     | $6.66 \times 10^{-10}$ | 31 | ×      | 18.18 | ×      |
| $f_5$ | 34000     | $3.43 \times 10^{-10}$ | 32 | 38     | 13.44 | 9.78  |
| $f_6$ | $4 \times 10^{10}$ | $9.49 \times 10^{-10}$ | 30 | ×      | 15.24 | ×      |
| $f_7$ | 1550      | $1.39 \times 10^{-10}$ | 33 | ×      | 28.17 | ×      |
| $f_8$ | 21.20     | $7.62 \times 10^{-10}$ | 31 | 42     | 28.16 | 98.30 |
| $f_9$ | 830       | $8.53 \times 10^{-10}$ | 30 | 45     | 17.42 | 9.71  |
| $f_{10}$ | 830     | $8.60 \times 10^{-10}$ | 30 | 44     | 20.73 | 25.47 |
| $f_{11}$ | 65      | $3.47 \times 10^{-10}$ | 32 | 32     | 18.94 | 8.33  |
| $f_{12}$ | $2.8 \times 10^6$ | $1.98 \times 10^{-10}$ | 33 | 36     | 27.11 | 7.44  |
| $f_{13}$ | 50       | $7.89 \times 10^{-10}$ | 31 | 61     | 13.29 | 10.81 |
| $f_{14}$ | 22.20    | $1.24 \times 10^{-10}$ | 33 | 34     | 45.37 | 99.04 |
| $f_{15}$ | 990      | $1.67 \times 10^{-10}$ | 33 | 43     | 23.45 | 17.72 |
| $f_{16}$ | 730      | $9.76 \times 10^{-10}$ | 30 | 48     | 29.51 | 21.84 |
| $f_{17}$ | 1250     | $3.66 \times 10^{-10}$ | 32 | ×      | 35.44 | ×      |
| $f_{18}$ | 900      | $4.96 \times 10^{-10}$ | 32 | ×      | 19.54 | ×      |
| $f_{19}$ | 850      | $1.55 \times 10^{-10}$ | 33 | ×      | 11.38 | ×      |
| $f_{20}$ | 880      | $3.78 \times 10^{-10}$ | 32 | ×      | 15.31 | ×      |
| $f_{21}$ | 1370     | $7.18 \times 10^{-10}$ | 31 | 35     | 19.38 | 12.32 |
| $f_{22}$ | 1450     | $4.25 \times 10^{-10}$ | 32 | 36     | 16.38 | 17.65 |
| $f_{23}$ | 1360     | $5.67 \times 10^{-10}$ | 32 | 36     | 12.79 | 8.16  |
| $f_{24}$ | 1430     | $3.61 \times 10^{-10}$ | 32 | 45     | 29.53 | 74.87 |

Fig. 4 Evolutions of SADE-L, ADAM, and ISp algorithms for function $f_{10}$ at 50-dimension.

optimum results given an ample number of splitting levels.

References

[1] H. Kahn and T. E. Harris, Estimation of particle transmission by random sampling, *National Bureau of Standards Applied Mathematics Series*, vol. 12, pp. 27–30, 1951.

[2] Z. I. Botev, An algorithm for rare-event probability estimation using the product rule of probability theory, https://www.researchgate.net/publication/43492609, 2008.

[3] W. S. Wadman, M. S. Squillante, and S. Ghosh, Accelerating splitting algorithms for power grid reliability estimation, in *Proc. 2016 Winter Simulation Conf.*, Washington, DC, USA, 2016, pp. 1757–1768.

[4] H. Louvin, E. Dumonteil, T. Lelièvre, M. Rousson, and C. M. Diop, Adaptive multilevel splitting for Monte Carlo particle transport, *EPJ Web of Conferences*, vol. 153, p. 06006, 2017.

[5] L. P. Wang and W. H. Fan, A multi-level splitting algorithm based on differential evolution, *Int. J. Model. Simul. Sci. Comput.*, vol. 9, no. 2, p. 1850021, 2018.

[6] J. Morio, R. Pastel, and F. Le Gland, An overview of importance splitting for rare event simulation, *Eur. J. Phys.*, vol. 31, no. 5, pp. 1295–1303, 2010.

[7] C. E. Bréhier and T. Lelièvre, On a new class of score functions to estimate tail probabilities of some stochastic processes with adaptive multilevel splitting, *Chaos*, vol. 29, no. 3, p. 033126, 2019.

[8] D. Aristoff, T. Lelièvre, C. G. Mayne, and I. Teo, Adaptive multilevel splitting in molecular dynamics simulations, *ESAIM: Proceedings and Surveys*, vol. 48, pp. 215–225, 2015.

[9] X. S. Suo, X. Q. Yu, and H. S. Li, Subset simulation for
multi-objective optimization, *Appl. Math. Model.*, vol. 44, pp. 425–445, 2017.

[10] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari, *Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization*. Singapore: Nanyang Technological University, 2005.

**Wenhui Fan** received the PhD degree from Zhejiang University in 1998. He is currently a professor at the Department of Automation and National CIMS Engineering Research Center, Tsinghua University, Beijing, China. His research interests include multidisciplinary collaborative design, simulation, and optimization.

**Liping Wang** received the BEng degree from Changchun University of Technology in 2006, and the MEng degree from Artillery Academy of PLA in 2009. She is a PhD candidate at Tsinghua University now. Her research interests include simulation of rare event and system engineering.