Differential Evolution Algorithm with Tracking Mechanism and Backtracking Mechanism

CUI, Laizhong; HUANG, Qiuling; LI, Genghui; YANG, Shu; MING, Zhong; WEN, Zhenkun; LU, Nan; LU, Jian

Published in:
IEEE Access

Published: 01/01/2018

Document Version:
Final Published version, also known as Publisher’s PDF, Publisher’s Final version or Version of Record

Publication record in CityU Scholars:
Go to record

Published version (DOI):
10.1109/ACCESS.2018.2864324

Publication details:

Citing this paper
Please note that where the full-text provided on CityU Scholars is the Post-print version (also known as Accepted Author Manuscript, Peer-reviewed or Author Final version), it may differ from the Final Published version. When citing, ensure that you check and use the publisher's definitive version for pagination and other details.

General rights
Copyright for the publications made accessible via the CityU Scholars portal is retained by the author(s) and/or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights. Users may not further distribute the material or use it for any profit-making activity or commercial gain.

Publisher permission
Permission for previously published items are in accordance with publisher’s copyright policies sourced from the SHERPA RoMEO database. Links to full text versions (either Published or Post-print) are only available if corresponding publishers allow open access.

Take down policy
Contact lbscholars@cityu.edu.hk if you believe that this document breaches copyright and provide us with details. We will remove access to the work immediately and investigate your claim.
Differential Evolution Algorithm With Tracking Mechanism and Backtracking Mechanism

LAIZHONG CUI1, QIULING HUANG1, GENGHUI LI1,2, SHU YANG1, ZHONG MING1, ZHENKUN WEN1, NAN LU1, AND JIAN LU3

1College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China
2Department of Computer Science, City University of Hong Kong, Hong Kong
3College of Mathematics and Statistics, Shenzhen University, Shenzhen 518060, China

Corresponding author: Genghui Li (li_genghui@126.com and genghuili2-c@my.cityu.edu.hk)

This work was supported in part by the National Natural Science Foundation of China under Grant 61772345, Grant 61402294, and Grant 61672358, in part by the Major Fundamental Research Project in the Science and Technology Plan of Shenzhen under Grant JCJY20150332410036842, Grant JCJY20160310095523765, and Grant JCJY20160307111232895.

ABSTRACT  Differential evolution (DE) is a simple and effective evolutionary algorithm that can be used to solve various optimization problems. In general, the population of DE tends to fall into stagnation or premature convergence so that it is unable to converge to the global optimum. To solve this issue, this paper proposes a tracking mechanism (TM) to promote population convergence when the population falls into stagnation and a backtracking mechanism (BTM) to re-enhance the population diversity when the population traps into the state of premature convergence. More specifically, when the population falls into stagnation, the TM is triggered so that the individuals who fall into the stagnant situation will evolve toward the excellent individuals in the population to promote population convergence. When the population goes into the premature convergence status, the BTM is activated so that the premature individuals go back to one of the previous statuses so as to restore the population diversity. The TM and BTM work together as a general framework and they are embedded into six classic DEs and nine state-of-the-art DE variants. The experimental results on 30 CEC2014 test functions demonstrate that the TM and BTM are able to effectively overcome the issues of stagnation and premature convergence, respectively, and therefore, enhance the performance of the DE significantly. Moreover, the experimental results also verify that the TM works together with the BTM as a general framework is better than other similar general frameworks.

INDEX TERMS Backtracking mechanism, differential evolution, premature convergence, stagnation, tracking mechanism.

I. INTRODUCTION

The main methods to deal with the optimization problems can be divided into the traditional optimization methods [1]–[4] and the heuristic optimization methods. The traditional optimization methods generally are suitable for solving the optimization problems with some good mathematical properties, such as continuity, derivative, convexity, and so on [2]. However, with the continuous development of science and technology, the optimization problems in engineering and practical applications often do not have these properties. Therefore, the heuristic optimization methods are designed by researchers to solve the increasingly complex optimization problems [5]–[8]. The current mainstream heuristic optimization algorithms include genetic algorithm (GA) [9], artificial bee colony algorithm (ABC) [10]–[15], differential evolution algorithm (DE) [16]–[22], particle swarm optimization algorithm (PSO) [23], and so on. Among them, DE is a very simple and effective heuristic optimization algorithm, which was first proposed by Storn and Price [24] in 1995 to solve real number optimization problems, and it has been used to deal with a variety of practical engineering optimization problems [25]–[27].

DE is a population-based stochastic optimization algorithm, which mainly consists of three core evolutionary operators, mutation, crossover and selection. In detail, for each individual (target vector/parent) in the population, first, the mutation operator is used to produce a mutation vector (donor vector), and then the crossover operator is employed to restructure the genetic material (gene-variable) of the target vector and the mutation vector for generating a trial vector.
promising regions. By this greedy guiding way, the population could be able to converge to a promising region. On the other hand, if the population traps into premature convergence status, a backtracking mechanism (BTM) will be triggered to make the population escape from this status and re-enhance the population diversity. In BTM, the individual will trace back to a previous position so as to the population diversity can be restored, and the population can be able to escape from local optima. In summary, the contributions of this paper are summarized as follows:

1) A tracking mechanism (TM) is proposed to solve the issue of stagnation. Specifically, when the individual falls into stagnation, the stagnant individual is forced to learn from the excellent individuals in the current population. TM can enhance the greedy nature of evolution and enable the population to converge to the currently promising regions.

2) A backtracking mechanism (BTM) is proposed to solve the issue of premature convergence. When the population is in the premature convergence status, the backtracking mechanism forces the individual to go back to one of its previous certain status and start to evolve again.

The rest of the paper is organized as follows. Section II briefly introduces the basic DE algorithm and related works. Section III makes a detail description to stagnation and premature convergence of DE. Section IV elaborates our proposed track mechanism and backtracking mechanism. Section V provides the numerical experiments and analysis. Finally, Section VI concludes this paper.

II. DE AND RELATED WORK

A. DE

1) POPULATION INITIALIZATION

DE starts from an initial population to evolve, and the population is composed of NP D-dimensional individuals \( X_{i,G} = (x_{i,1,G}, x_{i,2,G}, \cdots, x_{i,D,G}) \). Each individual of them is usually randomly generated from the search space \( S = \prod_{j=1}^{D} [x_{j}^{min} - x_{j}^{max}] \), where \( G \) is the number of iterations, \( NP \) is the population size, \( D \) is the dimension size, \( X^{min} \) and \( X^{max} \) are the low and upper bound, respectively. After the generation of the initial population, DE evolves the population from generation to generation by the mutation operator, crossover operator and selection operator.

2) MUTATION

Mutation operator is used to generate the mutation vector \( V_{i,G} = (v_{i,1,G}, v_{i,2,G}, \cdots, v_{i,D,G}) \) for the individual (target vector/ parent) of the population. Here, six basic and most commonly used mutation operators are given below.

1) DE/rand/1

\[
V_{i,G} = X_{i,G} + F \cdot (X_{r_2,G} - X_{r_3,G})
\]  

2) DE/rand/2

\[
V_{i,G} = X_{i,G} + F \cdot (X_{r_2,G} - X_{r_3,G}) + F \cdot (X_{r_4,G} - X_{r_5,G})
\]
is still retained. Mathematically, the selection operator is defined as follows. If the fitness value is less than or equal to the fitness value $f_i$, the trial vector $u_i$ is selected by using the neighbor information. Otherwise, the target vector $X_i$ is still retained. Mathematically, the selection operator is defined as follows.

$$X_{i,G+1} = \begin{cases} U_{i,G} & \text{if } f(U_{i,G}) \leq f(X_{i,G}) \\ X_{i,G} & \text{otherwise} \end{cases}$$

Intuitively, in the evolutionary process of DE, the offspring will be generated for the target vectors by mutation operator and crossover operator, and the selection operator makes the population keep good individuals and eliminate the inferior individual so as to lead the search process towards the global optimal solution.

### 2) THE GENERAL FRAMEWORKS

Moreover, many extensible general frameworks have been proposed for DE. In Pro-DE [53], the parents of the mutation operation are selected by using the neighbor information. In particular, the target individual tends to select the individuals that are close to itself for generating its mutation vector. Generally, the good individuals always consist of good genes. Therefore, to fully make use of these gene material, in rank-DE [27], the better fitness will have the higher ranking, and thus the superior individuals always have higher probability to generate offspring. Moreover, in NDi-DE [54], a neighbor guided selection mechanism (NGS) and a directed induced mutation strategy (DIM) were proposed to generate the mutation vector by cooperatively considering the neighbor and direction information. This framework is able to balance the exploration and exploitation effectively. Similarly, DMDE [55] also improved the performance of DE by identifying the good variation direction. Furthermore, by introducing the impulsive control (IPC) and event-triggered mechanism (ETM) into the DE algorithm, a new event-triggered impulsive (ETI) framework was formed for DE [56]. In detail, at the end of each generation, the ETI mechanism is triggered when the population’s update rate drops or becomes to be zero. NaDE established a general framework [57], which consists of an adaptive selection of neighborhood topology structure (ASNT) based on the index.

### 3) CROSSOVER

The crossover operator is applied to generate the trial vector (offspring) $U_{i,G} = (u_{i,1,G}, u_{i,2,G}, \cdots, u_{i,D,G})$ for the target vector $X_{i,G}$ by recombining the gene of the target vector and the mutation vector. Generally, the commonly used binomial crossover is mathematically defined as follows.

$$u_{i,j,G} = \begin{cases} v_{i,j,G} & \text{if } \text{rand}(0, 1) \leq Cr \text{ or } j = j_{\text{rand}} \\ X_{i,j,G} & \text{otherwise} \end{cases}$$

where $j_{\text{rand}}$ is a positive integer randomly selected in $[1, 2, \cdots, D]$ and $Cr$ is the crossover rate.

### 4) SELECTION

In the selection operator, for the minimization problems, if the fitness value $f(U_{i,G})$ of the trial vector $U_{i,G}$ is less than or equal to the fitness value $f(X_{i,G})$ of the target vector $X_{i,G}$, the trial vector $U_{i,G}$ will replace the target vector $X_{i,G}$ and enter the population. Otherwise, the target vector $X_{i,G}$ is still retained. Mathematically, the selection operator is defined as follows.

$$V_{i,G} = X_{\text{best},G} + F \cdot (X_{r_1,G} - X_{r_2,G})$$

### 5) DE/current-to-rand/1

$$V_{i,G} = X_{i,G} + F \cdot (X_{r_1,G} - X_{r_2,G}) + F \cdot X_{r_3,G}$$

### 6) DE/current-to-best/1

$$V_{i,G} = X_{i,G} + F \cdot (X_{\text{best},G} - X_{r_1,G}) + F \cdot (X_{r_2,G} - X_{r_3,G})$$

$X_{r_i,G}$ is an individual randomly selected in the population, where $i \neq r_1 \neq r_2 \neq \cdots \neq X_{\text{best},G}$ is the best individual found so far. $F$ is the scaling factor.

### 2) THE GENERAL FRAMEWORKS

Moreover, many extensible general frameworks have been proposed for DE. In Pro-DE [53], the parents of the mutation operation are selected by using the neighbor information. In particular, the target individual tends to select the individuals that are close to itself for generating its mutation vector. Generally, the good individuals always consist of good genes. Therefore, to fully make use of these gene material, in rank-DE [27], the better fitness will have the higher ranking, and thus the superior individuals always have higher probability to generate offspring. Moreover, in NDi-DE [54], a neighbor guided selection mechanism (NGS) and a directed induced mutation strategy (DIM) were proposed to generate the mutation vector by cooperatively considering the neighbor and direction information. This framework is able to balance the exploration and exploitation effectively. Similarly, DMDE [55] also improved the performance of DE by identifying the good variation direction. Furthermore, by introducing the impulsive control (IPC) and event-triggered mechanism (ETM) into the DE algorithm, a new event-triggered impulsive (ETI) framework was formed for DE [56]. In detail, at the end of each generation, the ETI mechanism is triggered when the population’s update rate drops or becomes to be zero. NaDE established a general framework [57], which consists of an adaptive selection of neighborhood topology structure (ASNT) based on the index.
and a neighborhood-dependent directional mutation operator (NDDM). Besides, in Rcr-DE [26], a repaired crossover rate record method can also be used as a general way for adjusting the crossover rate of crossover operator.

3) NEW SELECTION MECHANISM

Environmental selection operator imitates the rule of the fittest of the survival, which is also very important to the performance of DE. Most of DE methods used the fitness-based parent-offspring pair comparison method. However, to balance the exploration and exploitation, some other selection methods have been proposed. In IPDE [58], a new selection mechanism named information preserving (IP) strategy was proposed to enhance the ability of DE, in which fitter individuals among all target vectors and trial vectors will enter the next generation. In ESADE [59], a selection mechanism based on the idea of simulated annealing algorithm (SA) was proposed to retain some of the promising trial vectors that do not have very good fitness but have good diversity contribution. In GSDE [60], a particular selection mechanism was designed, in which inferior individuals have greater probability of being eliminated and replaced with a newly generated individual. In IDEI [34], a weighted fitness function calculation method that combines the relative size of the fitness value and the relative position of individual was designed for the selection. The experimental results show that this new selection is also able to improve the performance of DE on some optimization problems.

The literature review in here cannot cover all related works of DE, and more works can be referred to the survey of DE [61]. This paper aims to propose a new mechanism to solve the issues of stagnation and premature convergence.

III. STAGNATION AND PREMATURE CONVERGENCE OF DE

As shown in Fig. 1, DE starts to evolve from an initial population. After evolution through generation and generation, the population will enter three different statuses [62]: 1) Correct convergence status, which indicates the population converges to the global optimal area; 2) Premature convergence status, which means the population converges to the local optimal area; 3) Stagnation status. To employ suitable strategy to make the population break away premature convergence and stagnation, the key point is how to identify these different population statuses. For better understanding, we first use the basic DE (DE/rand/1 with NP=100, F = 0.5, Cr=0.9) to solve three 30-dimensional CEC2014 test problems, and the function error curves and population diversity curves are illustrated in Fig. 2, where the definitions of the function error and population diversity [34] are given as follows:

$$\text{error}_G = f\left(X_{\text{best},G}\right) - f\left(X^{*}\right)$$  \hspace{1cm} (9)

$$\text{div}_G = \frac{1}{NP} \sum_{i=1}^{NP} \left| X_{i,G} - \frac{1}{NP} \sum_{i=1}^{NP} X_{i,G} \right|$$  \hspace{1cm} (10)

where $X_{\text{best},G}$ is the best individual found so far, $X^{*}$ is the global optimal solution, $NP$ is the population size, $\left| X_{i,G} - X_{k,G} \right|$ is the Euclidean distance between $X_{i,G}$ and $X_{k,G}$ in the search space. Regarding F6, as shown in Fig. 2 (a), with the evolution of the population, the population diversity and function error gradually decrease to zero simultaneously, which indicates that the population converges correctly to the global optimal area. Regarding F14, as shown in Fig. 2 (b), the function error and population diversity no longer drop from about 100th generation. But in this stable status, the population diversity is greater than zero, which indicates that the population falls into the stagnation status from about the 100th generation. For F23, as shown in Fig. 2 (c), from about the 1700th generation, the population diversity nearly decreases to be 0, but the function error values is larger than 0, which means that the population falls into a local optima so as to result in the premature convergence. Note that in the real optimization environment, the real global optimal solution $X^{*}$ is unknown. Therefore, we will use the consecutive unsuccessful update and population diversity to identify the stagnation and premature statuses, the details of which will be described in the next subsection.

A. STAGNATION OF DE

Stagnation means that the population is unable to produce better solutions [40], [41]. In this paper, we use the consecutive unsuccessful update $CU_i (i = 1, 2, \cdots, NP)$ to identify the stagnant individual, and it can be mathematically defined as follows:

$$CU_i = \begin{cases} 
0, & \text{if } f\left(U_{i,G}\right) < f\left(X_{i,G}\right) \\
CU_i + 1, & \text{otherwise} 
\end{cases} \hspace{1cm} (11)$$

where $CU_i$ denotes the consecutive unsuccessful update of the $i$th individual in the population, and it is initialized to 0. In detail, at each generation, if the trial vector is better than the target vector, its $CU_i$ will be set to 0. Otherwise, its $CU_i$ will be increased by 1.
where $\varepsilon$ is also a user-defined threshold, and $PC_i = 1$ indicates that the $i$th individual has fallen into the premature convergence status. The specific analysis of the parameter $\varepsilon$ is given in the experimental part.

### B. PREMATURE CONVERGENCE OF DE

Premature convergence is a special stagnation status, in which almost all individuals in the population are stagnant and the entire population has converged to a small areas or a fixed point. Some definitions of premature convergence at variable level [39] and individual level [62] have been proposed. In this paper, we use the definition at individual level, and it is defined as follows,

$$PC_i = \begin{cases} 
1, & ST_i = 1 \&\& \text{div}_{G+1} \leq \varepsilon \\
0, & \text{otherwise} 
\end{cases}$$

where $\varepsilon$ is a user-defined threshold and $ST_i = 1$ means the $i$th individual is stagnant. In this paper, the user-defined threshold is defined as $Q = \lambda \times D$, where $D$ is the dimension of the search space. The empirical analysis to the parameter $\lambda$ will be given in the experimental part.

### IV. TRACKING AND BACKTRACKING FRAMEWORK

#### A. TRACKING MECHANISM

To deal with the issue of stagnation and make the population further evolve, in this paper, we propose a tracking mechanism (TM). In summary, TM aims at breaking away the stagnation status by making the current stagnant individuals learn from the superior individuals so that the population could be able to converge to a current promising area. For this purpose, TM defines a new selection mechanism. The pseudo-code of TM selection mechanism is given in Algorithm 1.

As shown in Algorithm 1, in TM selection mechanism, the original selection operator (line 1-12) of DE is kept except that it exchanges the value of the target vector ($X_{i,G}$) and the trial vector ($U_{i,G}$) when the trial vector is not worse than the target vector (line 8). After the original selection operator, TM is carried out, where the stagnant individuals will be replaced by the fitter individuals saved in the trial population (line 15). In this manner, the advantages lie in two aspects: 1) The stagnant individuals can be shifted into new better positions so as to pay more attention to search in the promising areas and enhance exploitation; 2) The competition extends from parent-offspring to individual-individual, and this is more suitable for the laws of natural evolution. But note that if all individuals saved in the trial population $U$ are worse than the current stagnant target vector, TM will not be carried out for this stagnant individual.

### Algorithm 1 TM Selection Mechanism

**Input:** Target population $P$, trial population $U$, $CU$,

1: for $i = 1 : NP$
2:   if $f(U_{i,G}) \leq f(X_{i,G})$
3:     $CU_i = 0$
4:   else
5:     $CU_i = CU_{i+1} + 1$
6:     end if
7:   end for
8: $X_{i,G+1} = U_{i,G}$, $U_{i,G} = X_{i,G}$
9: end if
10: $X_{i,G+1} = X_{i,G}$, $CU_i = CU_{i+1}$
11: end if
12: end for

**Output:** Target $P$, $CU$

### B. BACKTRACKING MECHANISM

In some special situations, on one hand, the population may directly fall into the premature convergence status. On the other hand, TM mechanism may also cause the population is trapped into the premature convergence status. To solve this issue, this paper further proposes a backtracking mechanism (BTM). In summary, BTM aims at making the population escape from the premature convergence by letting the current premature individuals go back to their previous positions so that the population diversity can be restored. To this end, BTM also establishes a new selection mechanism. The pseudo-code of BTM selection mechanism is described in Algorithm 2.

As shown in Algorithm 2, in BTM selection mechanism, the original selection operator (line 1-16) of DE is also
into most of DE variants by modifying the selection operator. On the basis of above TM and BTM, we propose a new MECHANISM to save the evolution trajectory of the best individual found so far.

Algorithm 2 BTM Selection Mechanism

Input: Target population P, trial population U, CU, PC, \( A_i(i = 1, \cdots, NP), k \)

1: for \( i = 1 : NP \)
2: \( \text{if} \ f(U_{i,G}) \leq f(X_{i,G}) \)
3: \( \text{if} f(U_{i,G}) < f(X_{i,G}) \)
4: if \( \text{size}(A_i) == k \)
5: \( \text{Delete the oldest individual of archive} \ A_i \),
6: end if
7: \( \text{Put} \ X_{i,G} \ \text{into archive} \ A_i \) as the latest individual.
8: \( CU_i = 0 \)
9: else
10: \( CU_i = CU_i + 1 \)
11: end if
12: \( X_{i,G+1} = U_{i,G} \)
13: else
14: \( X_{i,G+1} = X_{i,G}, CU_i = CU_i + 1 \)
15: end if
16: end for
17: Calculate population \( \text{div}_{G+1} \) (Eq. (10)).
18: for \( i = 1 : NP \)
19: if \( PC_i == 1 \ & \ & X_{i,G+1} \neq X_{\text{best},G+1} \)
20: \( \text{Set} \ X_{i,G+1} \ \text{as the latest individual of archive} \ A_i \).
21: \( CU_i = 0 \)
22: \( \text{Delete the latest individual of archive} \ A_i \).
23: end if
24: end for

Output: Target P, CU, A\(_i\)(i = 1, \cdots, NP)

retained excluding that the target vector \( (X_{i,G}) \) will be saved in its external archive \( (A_i) \) when a better offspring is found (line 7). The external archives \( A_i(i = 1, \cdots, NP) \) are used to save the evolution trajectory of the \( i \text{th} \) individual of the population. Similarly, after the original selection operator, BTM is executed, in which if the population diversity is small enough, the premature individuals (line 20) will be replaced by the best individuals (latest individual) saved in the corresponding external archives and the best individual (latest individual) will be deleted from the archive (line 20-22). But, note that since the current best individual represents the current most promising area, in order not to lose this area, BTM is not applied to the best individual found so far.

It is noteworthy that the archive for each individual is a sliding window with fixed size \( k \) (\( k \) is set to 50 in our experiment), and the first in first out rule is used to manage the archive overflow.

C. DIFFERENTIAL EVOLUTION ALGORITHM WITH TRACKING MECHANISM AND BACKTRACKING MECHANISM

On the basis of above TM and BTM, we propose a new general framework (TBT) for DE, which can be embedded into most of DE variants by modifying the selection operator.

The complete pseudo-code of TBT-DE algorithm is provided in Algorithm 3. As shown in Algorithm 3, the first step is to set the necessary parameters and generate an initial
population (line 1). After initialization, TBT-DE goes into a loop of mutation, crossover and selection. Mutation and crossover (line 3-6) can be kept the same as any DE methods. The difference between TBT-DE and other DE methods relies on the selection (line 8-43), which includes the original selection operator (line 9 and 19-22), TM (line 19 and 32) and BTM (line 11-14, 28-31 and 38-40). Intuitively, TBT-DE increases the computational complexity to a certain extent and need additional memory.

The computational complexity of the basic DE/rand/1 is $O(G_{\text{max}} \cdot \mathcal{NP} \cdot D)$. For TBT-DE/rand/1, first, the additional budget for TM mechanism relies on choosing the tracked individual (line 10 in Algorithm 1), and its computational complexity is $O(\mathcal{NP}^2)$. Second, the extra overhead of BTM depends on selecting the best individual from the external archive, and the complexity of this procedure is $O(1)$ since the individuals in archive were sorted based on fitness (line 17 in Algorithm 2). Therefore, the computational complexity of BTM is $O(\mathcal{NP})$. Overall, the computational complexity of TBT-DE is $O(G_{\text{max}} \cdot \mathcal{NP} \cdot D + G_{\text{max}} \cdot \mathcal{NP}^2)$, and it can be simplified as $O(G_{\text{max}} \cdot \mathcal{NP} \cdot D)$ when $\mathcal{NP} < D$ or $O(G_{\text{max}} \cdot \mathcal{NP}^2)$ when $\mathcal{NP} > D$.

In fact, our proposed method uses some external archives to solve the stagnation and premature issues, which is very similar to the archive mechanism of JADE [29]. The essential differences between TBT-DE and JADE are summarized as follows: 1) JADE keeps an archive for all population members, while TBT-DE establishes an independent archive for each population member. 2) JADE uses the archive to keep population diversity and provide promising evolution directions. However, excluding providing population diversity, TBT-DE makes use of archives to record the evolution tracks of each population member, which is able to make population member backtrack to the previous positions so as to avoid the premature issue. 3) The mechanisms of handling archive overflow are also different. JADE randomly deletes the redundant individuals, while TBT-DE obeys the first in first out rule to remove the oldest individual.

Moreover, AEPD-DE [39] also aims to solve the issues of population stagnation and premature convergence, but there are some essential differences. On the one hand, the definitions of stagnation and premature convergence are distinct. AEPD-DE determines stagnation and premature convergence based on dimension level. In detail, the dimension is stagnant when the number of un-successive generations of this dimension is larger than or equal to a threshold that is proportional to the population size. And the dimension is converged when the population diversity of this dimension is smaller than or equal to an adaptive threshold. In general, AEPD mechanism is triggered when all dimensions are stagnant or converged. Besides, the proactive trigger of AEPD mechanism may also be executed with a small probability. However, TBT-DE determines stagnation and premature convergence based on individual level. Elaborately, the individual is stagnant when the number of consecutive unsuccessful updates of this individual is larger than or equal to a threshold that is proportional to the dimension size. And the individual falls into premature convergence when this individual is stagnant and the population diversity is smaller than or equal to a fixed threshold. The trigger of TBT mechanism only requires a single individual to be stagnant or premature convergence. On the other hand, the ways of handling stagnation and premature convergence are also different. AEPD-DE simultaneously solves stagnation and premature convergence by generating new individuals to enhance population diversity, while TBT-DE separately handles stagnation and premature convergence through using tracking and backtracking mechanism that exploit the present and previous individuals, respectively.

V. EXPERIMENTAL RESULT
In this section, we will elaborate the numerical experiments to demonstrate the efficiency and effectiveness of our proposed framework. In detail, Section V.A introduces the experimental setup. Section V.B demonstrates the effectiveness of the TBT framework by combining it with six basic DE algorithms. Section V.C compares TBT with other two general frameworks. Section V.D studies the performance of TBT on some state-of-the-art DE variants. Section V.E analyzes the behavior of TM and BTM. Finally, Section V.F makes a sensitiveness analysis of the control parameters. Section V.G discusses how to set the stagnation threshold. Section V.H discusses how to set the backtracking step size. Section V.I compares the runtime of DEs and corresponding TBT-DEs.

A. EXPERIMENTAL SETUP
To study the performance of the proposed framework, 30 benchmark functions with $D = 30$, $D = 50$ and $D = 100$ derived from CEC2014 single objective optimization competition [63] are employed. Among these 30 functions, F1-F3 are unimodal functions, F4-F16 are simple multimodal functions, F17-F22 are hybrid functions, and F23-F20 are composition functions. The details about these benchmark can be referred to [63]. For a fair comparison, the maximum number of function evaluations is adopted as the stop criteria. Each algorithm conducts 30 independent runs on each function, which is set to 10000D according to the guideline provided by CEC2014 competition. For TBT, the necessary parameters are roughly set as $\lambda = 1.5$, $\varepsilon = 10^{-4}$ in the following comparative experiments.

To analyze the experimental result in a statistical meaning way, the Wilcoxon rank-sum test at a 5% significant level is conducted on each function, and the test results are denoted as $+/-/\sim$, which means the TBT-DE method is better than/ similar to/ worse than the corresponding DE method. Moreover, the Wilcoxon signed rank test is carried out on all functions to test the difference between the performance of TBT-DE and corresponding DE. The test results are denoted by R+ value/ R− value, which means that the rank sum of the TBT-DE is better/worse than the corresponding compared algorithm. In addition, the p-values are used to show the significant difference. In detail, if the p-value is smaller than the
TABLE 1. Comparison among the six basic DE algorithms and the corresponding TBT-DE variants ($D=30$).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$+/=/-$</th>
<th>$R^+$</th>
<th>$R^-$</th>
<th>$p$-value</th>
<th>$\alpha = 0.05$</th>
<th>$\alpha = 0.1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBT-DE/rand/1 vs. DE/rand/1</td>
<td>17/12/1</td>
<td>392.0</td>
<td>73.0</td>
<td>9.99E-04</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>TBT-DE/rand/2 vs. DE/rand/2</td>
<td>30/0/0</td>
<td>465.0</td>
<td>0.0</td>
<td>2.00E-06</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>TBT-DE/best/1 vs. DE/best/1</td>
<td>1/27/2</td>
<td>229.0</td>
<td>236.0</td>
<td>9.34E-01</td>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td>TBT-DE/best/2 vs. DE/best/2</td>
<td>12/18/0</td>
<td>328.0</td>
<td>107.0</td>
<td>1.64E-02</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>TBT-DE/current-to-rand/1 vs. DE/current-to-rand/1</td>
<td>10/20/0</td>
<td>271.0</td>
<td>194.0</td>
<td>4.22E-01</td>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td>TBT-DE/current-to-best/1 vs. DE/current-to-best/1</td>
<td>4/24/2</td>
<td>204.0</td>
<td>261.0</td>
<td>5.31E-01</td>
<td>=</td>
<td>=</td>
</tr>
</tbody>
</table>

significant level ($\alpha$), a significant difference has been tested. Otherwise, no significant difference has been found. We also use “+”, “=” and “−” to denote that the performance of TBT-DE is significantly “superior to”, “similar to” and “inferior to” the corresponding compared algorithm.

B. EFFECTIVENESS ON BASIC DE METHODS

In this section, we first embed TBT into six basic DE methods (i.e., DE/rand/1, DE/rand/2, DE/best/1, DE/best/2, DE/current-to-rand/1, and DE/current-to-best/1) to yield six corresponding TBT-DE methods (i.e., TBT-DE/rand/1, TBT-DE/rand/2, TBT-DE/best/1, TBT-DE/best/2, TBT-DE/current-to-rand/1, and TBT-DE/current-to-best/1). For all these DE methods and TBT-DE methods, the population size $NP$ is set to 100, scale factor $F$ is set to 0.5, and crossover rate $Cr$ is set to 0.9. The mean and standard deviation of function error on 30 $D$ functions are provided in the TABLE 1 and TABLE 2 (S. TABLEs 1 - 22 are seen in the supplemental file due to the space limitation). And the final statistical comparison results are provided in TABLE 1.

TABLE 1 shows that TBT-DE/rand/1, TBT-DE/rand/2, TBT-DE/best/2 are significantly better than the corresponding DE/rand/1, DE/rand/2 and DE/best/2, respectively. Moreover, TBT-DE/current-to-rand/1 shows very competitive performance with DE/current-to-rand/1 since it outperforms/underperforms DE/current-to-rand/1 on 10/0 function(s). Moreover, it gets higher $R^+$ value than $R^-$ value. But for TBT-DE/best/1 and TBT-DE/current-to-best/1, they get very similar results to the corresponding DE methods. The reason can be summarized that the DE/best/1 and DE/current-to-best/1 employ a very greedy mutation operator, which easily causes the population fall into the local optima. Although BTM mechanism is trigged to escape from the local optima, it always fails to produce a new solution that is better than the current best solution by using the DE/best/1 or DE/current-to-best/1 mutation operator, respectively. To demonstrate this statement, we provide the evolution behaviors of DE/best/1 and TBT-DE/best/1 on two representative functions in Fig. 3. As shown in Fig. 3, for F16, DE/best/1 easily goes into a local optimal, but TBT-DE/best/1 is able to escape from the premature convergence status and successfully find a better solution. However, for F25, although TBT-DE/best/1 can restore the population diversity by BTM mechanism when the population traps into a local optimum, it also cannot find a better solution since the population will enter the same local optimum quickly guided by the current best solution. Based on our preliminary experiments, on most functions, TBT-DE/best/1 and TBT-DE/current-to-best/1 are able to restore the population diversity, but they are unable to find a solution that is better than the current best solution. Therefore, TBT-DE/best/1 and TBT-DE/current-to-best/1 show similar performance with DE/best/1 and DE/current-to-best/1, respectively. In summary, TBT is able to improve the performance of DE/rand/1, DE/rand/2, DE/best/2 and DE/current-to-rand/1, but it is not useful for DE/best/1 and DE/current-to-best/1.

C. COMPARISON WITH SPS-DE AND GAR-DE

For dealing with stagnation and premature, other general frameworks (e.g., SPS-DE [25] and GAR-DE [26]) have been proposed. To show the different contributions with these general frameworks, we further compare TBT-DEs with SPS-DEs and GAR-DEs on 30 $30D$ functions. For SPS-DE methods, GAR-DE methods and TBT-DE methods, the population size $NP$, scale factor $F$ and crossover rate $Cr$ are set to 100, 0.5 and 0.9, respectively. The mean and standard deviation of function error are provided in the S. TABLE 3 to S. TABLE 6 (seen in the supplemental file). The statistical comparison results are given in TABLE 2 and TABLE 3, respectively.
As shown in TABLE 2, TBT-DE/rand/1 and TBT-DE/rand/2 are significantly better than the corresponding SPS-DE/rand/1 and SPS-DE/rand/2, respectively. TBT-DE/best/1, TBT-DE/best/2 and TBT-DE/current-to-best/1 are very competitive with the corresponding SPS-DEs. In detail, TBT-DE/best/1 outperforms/underperforms SPS-DE/best/1 on 2/4 functions, but TBT-DE/best/1 obtains higher R+ value than R− value. TBT-DE/best/2 is superior/inferior to SPS-DE/best/2 on 10/1 function(s). Similarly, TBT-DE/best/2 gets higher R+ value than R− value. TBT-DE/current-to-best/1 is significantly better/worse than SPS-DE/current-to-best/1 on 4/3 functions, but in this case SPS-DE/current-to-best/1 obtains lower R+ value. Moreover, at a significant level of 0.1, TBT-DE/current-to-rand/1 is
TABLE 8. Comparison between TBT-SWDE_success_mBLX and SWDE_success_mBLX.

<table>
<thead>
<tr>
<th>TBT-SWDE_success_mBLX vs. SWDE_success_mBLX</th>
<th>+/-/%-</th>
<th>R+</th>
<th>R-</th>
<th>p-value</th>
<th>$\alpha = 0.05$</th>
<th>$\alpha = 0.1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>D=30</td>
<td>9/17/4</td>
<td>266.0</td>
<td>169.0</td>
<td>2.89E-01</td>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td>D=50</td>
<td>9/17/4</td>
<td>224.0</td>
<td>211.0</td>
<td>8.80E-01</td>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td>D=100</td>
<td>8/21/1</td>
<td>263.5</td>
<td>171.5</td>
<td>3.15E-01</td>
<td>=</td>
<td>=</td>
</tr>
</tbody>
</table>

TABLE 9. Comparison between TBT-ZEPDE and ZEPDE.

<table>
<thead>
<tr>
<th>TBT-ZEPDE vs. ZEPDE</th>
<th>+/-/%-</th>
<th>R+</th>
<th>R-</th>
<th>p-value</th>
<th>$\alpha = 0.05$</th>
<th>$\alpha = 0.1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>D=30</td>
<td>5/9/16</td>
<td>97.0</td>
<td>368.0</td>
<td>5.04E-03</td>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td>D=50</td>
<td>4/15/11</td>
<td>192.0</td>
<td>273.0</td>
<td>3.98E-01</td>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td>D=100</td>
<td>3/17/10</td>
<td>151.0</td>
<td>314.0</td>
<td>9.17E-02</td>
<td>=</td>
<td>=</td>
</tr>
</tbody>
</table>

TABLE 10. Comparison between TBT-JADE_sort and JADE_sort.

<table>
<thead>
<tr>
<th>TBT-JADE_sort vs. JADE_sort</th>
<th>+/-/%-</th>
<th>R+</th>
<th>R-</th>
<th>p-value</th>
<th>$\alpha = 0.05$</th>
<th>$\alpha = 0.1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>D=30</td>
<td>4/20/6</td>
<td>143.5</td>
<td>321.5</td>
<td>6.56E-02</td>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td>D=50</td>
<td>5/21/4</td>
<td>177.0</td>
<td>258.0</td>
<td>3.75E-01</td>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td>D=100</td>
<td>6/22/2</td>
<td>288.0</td>
<td>147.0</td>
<td>1.25E-01</td>
<td>=</td>
<td>=</td>
</tr>
</tbody>
</table>

significantly worse than the corresponding SPS-DEs. In summary, TBT is better than SPS when considering all the cases.

It can be seen from TABLE 3, TBT-DE/rand/1, TBT-DE/rand/2, TBT-DE/best/1, and TBT-DE/best/2 are significantly better than the corresponding GAR-DE/rand/1, GAR-DE/rand/2, GAR-DE/best/1, and GAR-DE/best/2, respectively. For DE/current-to-rand/1 and DE/current-to-best/1, TBT shows similar performance with GAR, but in these two cases, TBT obtains lower R+ value. Therefore, TBT is better than GAR in most cases.

In summary, on the basis of these comparison experiments, we can conclude that TBT is better than SPS and GAR in most cases.

D. COMPARISON WITH ADVANCED DE ALGORITHM VARIANTS

TBT works as a general framework, and thus it can also be used to improve the performance of the state-of-the-art DE methods. To this end, we further embed TBT into nine state-of-the-art DE methods, i.e., JADE [29], CoDE [64], SHADE [65], L-SHADE [65], SWDE_success_mBLX [66], ZEPDE [67], JADE_sort [68], MSDE [69], and SaDE/Mexp [70], yielding nine corresponding TBT-DE methods. The parameter settings of these nine algorithms are kept the same as used in the original papers. Each algorithm and its corresponding TBT-based variant will conduct 30 independent runs on each function. The results of mean and standard deviation of function error are provided in S. TABLE 7 - S. TABLE 15 (seen in the supplemental file). The final statistical comparison results are given in TABLE 4 - TABLE 7.

From TABLE 4, TBT-JADE is significantly better than JADE in the cases of $D = 30$ and $D = 50$. In the case of $D = 100$, TBT-JADE is very competitive with JADE at a significant level of 0.05, but it is also significantly superior to JADE at a significant level of 0.1. Moreover, in all the cases, TBT-JADE obtains higher R+ values. In summary, TBT-JADE is able to significantly improve the performance of JADE.

As shown in TABLE 5, when $D = 50$ and $D = 100$, TBT-CoDE significantly outperforms CoDE. But when $D = 30$, TBT-CoDE and CoDE show similar performance. Moreover, TBT-CoDE gets lower R+ value than R-value. Therefore, TBT can remarkably improve the performance of CoDE on the medium ($D = 50$) and high ($D = 100$) dimension problems, but it slightly degrades the performance of CoDE on the low ($D = 30$) dimension problems.

From TABLE 6 in the case of $D = 100$, TBT-SHADE is significantly superior to SHADE. In the cases of $D = 30$ and $D = 50$, TBT-SHADE and SHADE exhibit competitive performance. Therefore, for SHADE, TBT is able to bring significantly positive effect on the high ($D = 100$) dimension problems. But it doesn’t remarkably affect the performance of SHADE on the low and medium dimension functions.

TABLE 7 - TABLE 10 show that TBT is not significantly affect the performance, compared with L-SHADE, ZEPDE, SWDE_success_mBLX, and JADE_sort. Moreover, in some cases, it may result in a heavily negative effect, such as for L-SHADE on 30D problems, ZEPDE on 30D and 100D problems, and JADE_sort on 30D problems.

As shown in TABLE 11 - TABLE 12, in all the cases ($D = 30$, $D = 50$ and $D = 100$), TBT significantly improves the performance of MSDE and SaDE/Mexp.

Overall, based on these comparison experiments, we can find out that although TBT is not able to always bring significantly positive effect on all state-of-the-art DE methods, it can make the improvement in most cases. Therefore, our
TABLE 11. Comparison between TBT-MSDE and MSDE.

<table>
<thead>
<tr>
<th>D</th>
<th>TBT-MSDE vs. MSDE</th>
<th>+/-</th>
<th>R+</th>
<th>R-</th>
<th>p-value</th>
<th>α = 0.05</th>
<th>α = 0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>17/11/2</td>
<td>377.0</td>
<td>58.0</td>
<td>5.41E-04</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>18/10/2</td>
<td>324.0</td>
<td>111.0</td>
<td>2.07E-02</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>14/16/0</td>
<td>354.5</td>
<td>110.5</td>
<td>1.17E-02</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
</tbody>
</table>

TABLE 12. Comparison between TBT-SaDE/Mexp and SaDE/Mexp.

<table>
<thead>
<tr>
<th>D</th>
<th>TBT-SaDE/Mexp vs. SaDE/Mexp</th>
<th>+/-</th>
<th>R+</th>
<th>R-</th>
<th>p-value</th>
<th>α = 0.05</th>
<th>α = 0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>14/16/0</td>
<td>350.5</td>
<td>114.5</td>
<td>1.48E-02</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>12/17/1</td>
<td>319.0</td>
<td>116.0</td>
<td>2.74E-02</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>12/16/2</td>
<td>343.5</td>
<td>121.5</td>
<td>2.18E-02</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
</tbody>
</table>

TABLE 13. Comparison between Different Mechanisms in SaDE/Mexp (D=30).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>+/-</th>
<th>R+</th>
<th>R-</th>
<th>p-value</th>
<th>α = 0.05</th>
<th>α = 0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>TM-SaDE/Mexp vs. SaDE/Mexp</td>
<td>14/15/1</td>
<td>339.5</td>
<td>125.5</td>
<td>2.70E-02</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>BTM-SaDE/Mexp vs. SaDE/Mexp</td>
<td>8/15/7</td>
<td>266.0</td>
<td>199.0</td>
<td>4.84E-01</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TBT-SaDE/Mexp vs. TM-SaDE/Mexp</td>
<td>1/27/2</td>
<td>243.5</td>
<td>191.5</td>
<td>5.67E-01</td>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td>TBT-SaDE/Mexp vs. BTM-SaDE/Mexp</td>
<td>9/20/1</td>
<td>307.5</td>
<td>127.5</td>
<td>5.04E-02</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

TABLE 14. Comparison between different values of parameters $\lambda$ in TBT-SaDE/Mexp (D=30).

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>+/-</th>
<th>R+</th>
<th>R-</th>
<th>p-value</th>
<th>α = 0.05</th>
<th>α = 0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda = 1$ vs. $\lambda = 1.5$</td>
<td>0/27/3</td>
<td>217.5</td>
<td>217.5</td>
<td>9.91E-01</td>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td>$\lambda = 1$ vs. $\lambda = 2$</td>
<td>0/25/5</td>
<td>159.5</td>
<td>275.5</td>
<td>2.06E-01</td>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td>$\lambda = 1$ vs. $\lambda = 5$</td>
<td>1/22/7</td>
<td>192.5</td>
<td>242.5</td>
<td>5.81E-01</td>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td>$\lambda = 1.5$ vs. $\lambda = 2$</td>
<td>1/28/1</td>
<td>105.5</td>
<td>329.5</td>
<td>1.50E-02</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\lambda = 1.5$ vs. $\lambda = 5$</td>
<td>1/24/5</td>
<td>199.0</td>
<td>266.0</td>
<td>4.84E-01</td>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td>$\lambda = 2$ vs. $\lambda = 5$</td>
<td>1/26/3</td>
<td>267.0</td>
<td>198.0</td>
<td>4.72E-01</td>
<td>=</td>
<td>=</td>
</tr>
</tbody>
</table>

E. ANALYSIS OF TWO MECHANISMS

TBT includes the tracking mechanism (TM) and the backtracking mechanism (BTM), and this subsection aims at analyzing the effectiveness of the tracking mechanism (TM) and backtracking mechanism (BTM) on SaDE/Mexp [70]. To this end, we provide the comparison results of four algorithms: 1) The original SaDE/Mexp, 2) SaDE/Mexp with TM, 3) SaDE/Mexp with BTM, 4) SaDE/Mexp with TBT. All these four algorithms execute 30 independent runs on 30 $30D$ functions. The results of mean and standard deviation of function error are provided in the S. TABLE 16 (seen in the supplemental file). The final statistical comparison results are summarized in Table 13.

From Table 13, TM-SaDE/Mexp is significantly better than SaDE/Mexp, which means TM is able to significantly improve the performance of SaDE/Mexp individually. BTM-SaDE/Mexp gets the similar performance with SaDE/Mexp, but BTM-SaDE/Mexp obtains higher R+ value than R− value, which also indicates BTM can also bring the contribution to SaDE/Mexp. Moreover, TBT-SaDE/Mexp performs significantly better than BTM-SaDE/Mexp at a significant level of 0.1, and it is similar to TM-SaDE/Mexp. In addition, the R+ values of TBT-SaDE/Mexp are higher than its R− values, which denotes that TBT-SaDE/Mexp is superior to TM-SaDE/Mexp and BTM-SaDE/Mexp. Therefore, we can draw a conclusion that both TM and BTM can make the contribution to SaDE/Mexp, and the contribution can be accumulated by combining TM and BTM. But we should highlight that the contribution of TM is always better than that of BTM. Moreover, to make a clear explanation, the convergence curve and population diversity curve of these four algorithms on some representative functions ($D = 30$) are illustrated in Fig. 4. As shown in Fig. 4, more insights can be given as follows: 1) For most of these functions, the population always falls into stagnant status (population diversity $\epsilon > 10^{-4}$), and TM mechanism can promote the population converge to a promising area for finding a better solution. Therefore, TM brings a significant contribution. 2) For BTM-SaDE/Mexp, on one hand, if the population is not trapped into a local optimum, the backtracking mechanism will not be triggered. On the other hand, even if the population falls into a local optima so that the backtracking mechanism is triggered,
it only provides more chances to explore other regions but it can’t be sure to find a better solution. This is why BTM cannot bring a significant contribution. 3) For TBT-SaDE/Mexp, TM is first exploited to promote the population converge to a promising area so that the backtracking mechanism can be triggered. In this manner, it not only retains the advantage of TM-SaDE/Mexp, but also restores the population diversity so as to provide the opportunity to search other areas.

**F. PARAMETER ANALYSIS**

TBT includes two new parameters, namely values \( \lambda \) and \( \varepsilon \). \( \lambda \) is used to control the threshold \( Q = \lambda \times D \) and judge whether an individual falls into stagnation status or not. Similarly, the threshold \( \varepsilon \) is used to judge whether the population is trapped into premature convergence status or not. In this subsection, the sensitiveness analysis experiments for these two parameters are conducted on TBT-SaDE/Mexp.

For parameter \( \lambda \), we test TBT-SaDE/Mexp with \( \lambda = 1 \), \( \lambda = 1.5 \), \( \lambda = 2 \) and \( \lambda = 5 \) on 30 30D functions. The results of mean and standard deviation of function error are provided in S. TABLE 17 (seen in the supplemental file). The final statistical comparison results are summarized in TABLE 14. As shown in TABLE 14, considering all functions, \( \lambda \) is insensitive to the performance of TBT-SaDE/Mexp. In general, \( \lambda = 2 \) is a promising setting.

Regarding parameter \( \varepsilon \), we test TBT-SaDE/Mexp with \( \varepsilon = 10^{-8} \), \( \varepsilon = 10^{-4} \), \( \varepsilon = 10^{-2} \) and \( \varepsilon = 1 \) on 30 30D functions. The results of mean and standard deviation of function error are given in S. TABLE 18 (seen in the supplemental file). The final statistical comparison results are summarized in TABLE 15. It can be seen that parameter \( \varepsilon \) does not remarkably affect the performance of SaDE/Mexp. \( \varepsilon = 10^{-2} \) is our recommended setting for SaDE/Mexp.

Finally, we should point out that TBT works as a general framework, and thus it may need different parameter settings when it is combined with different algorithms.

**G. DISCUSSION ON STAGNATION THRESHOLD**

The value of stagnation threshold \( Q \) is of great importance to determine the stagnation status. In general, the larger the dimension of the objective function is, the greater the search space is, and the more resources need to be used to explore. Therefore, \( Q \) should be positively related to the dimension size \( D \) [62]. In SPS-DE [40] and GAR-DE [41], \( Q \) was set to 32 and 60, respectively. However, in AMECoDEs [62], \( Q \) was set to the dimension size \( D \). To obtain a suitable threshold \( Q \) for landscapes with different search space in our proposed framework, we have tested some different settings for stagnation threshold \( Q \) as follows: 1) a constant value, i.e. \( Q = 32 \); 2) linear relation with dimension \( Q = D \) and \( Q = 1.5D \); 3) non-linear relation with dimension \( Q = D^2/25 \). The experiments are conducted on DE/rand/1 and the results of mean and standard deviation of function error for 30D, 50D, and 100D CEC2014 test problems are given in S.TABLE 19 - S. TABLE 21 (seen in the supplemental file). The experimental results are provided in TABLE 16 which demonstrates that \( Q = 1.5D \) is a very suitable setting for our framework.

**H. DISCUSSION ON BACKTRACKING STEP SIZE**

In TBT-DE, how to determine the backtracking step size is crucial since a large backtracking step sizes may result in wasting search resources while a small backtracking step sizes may make the search be unable to jump out the local
Here, we firstly compare two ways for setting the backtracking step, namely one-step backtracking and random backtracking. One-step backtracking means that the premature individual will backtrack to its previous best position (using the best individual of the archive to replace the current premature individual), and random backtracking represents that the premature individual will backtrack to a previous random position (randomly selecting an individual from the archive to replace the current premature individual). The experiments are conducted on SaDE/Mexp and the results of mean and standard deviation of function error for 30D CEC2014 test problems are given in S. TABLE 22 (seen in the supplemental file). The final statistical comparison results are summarized in TABLE 17. From TABLE 17, we can see that there is no significant difference between one-step backtracking and random backtracking. But one step backtracking is slightly better than random backtracking since it gets higher $R^+$ value than $R^-$ value. But we should point out that these experiments don’t aim to demonstrate that one-step backtracking is the best way for controlling the backtracking step size. How to effectively set the backtracking step size is still an open issue.

| TABLE 15. Comparison between different values of parameters $\varepsilon$ in TBT-SaDE/Mexp ($D=30$). |
|-----------------------------------|-----------------|-----------------|-----------------|-----------------|
| $\varepsilon=10^{-4}$ vs. $\varepsilon=10^{-4}$ | $+/=-$ | $R^+$ | $R^-$ | $p$-value | $\alpha=0.05$ | $\alpha=0.1$ |
| 0/26/4 | 231.5 | 203.5 | 7.54E-01 | - | - |
| $\varepsilon=10^{-4}$ vs. $\varepsilon=10^{-5}$ | $+/=-$ | $R^+$ | $R^-$ | $p$-value | $\alpha=0.05$ | $\alpha=0.1$ |
| 0/28/2 | 188.5 | 246.5 | 5.24E-01 | - | - |
| $\varepsilon=10^{-4}$ vs. $\varepsilon=1$ | $+/=-$ | $R^+$ | $R^-$ | $p$-value | $\alpha=0.05$ | $\alpha=0.1$ |
| 0/29/1 | 252.0 | 213.0 | 6.81E-01 | - | - |
| $\varepsilon=10^{-4}$ vs. $\varepsilon=10^{-2}$ | $+/=-$ | $R^+$ | $R^-$ | $p$-value | $\alpha=0.05$ | $\alpha=0.1$ |
| 0/29/1 | 185.0 | 280.0 | 3.24E-01 | - | - |
| $\varepsilon=10^{-4}$ vs. $\varepsilon=1$ | $+/=-$ | $R^+$ | $R^-$ | $p$-value | $\alpha=0.05$ | $\alpha=0.1$ |
| 1/28/1 | 216.5 | 218.5 | 9.74E-01 | - | - |
| $\varepsilon=10^{-2}$ vs. $\varepsilon=1$ | $+/=-$ | $R^+$ | $R^-$ | $p$-value | $\alpha=0.05$ | $\alpha=0.1$ |
| 0/29/1 | 275.5 | 159.5 | 2.06E-01 | - | - |

| TABLE 16. Comparison between different values of stagnation threshold $Q$ in TBT-DE/rand/1. |
|-----------------------------------|-----------------|-----------------|-----------------|-----------------|
| TBT-DE/rand/1 | Dimension | $+/=-$ | $R^+$ | $R^-$ | $p$-value | $\alpha=0.05$ | $\alpha=0.1$ |
| $Q=32$ vs. $Q=1.5D$ | 30D | 0/27/3 | 97.5 | 337.5 | 9.17E-03 | - | - |
| | 50D | 1/23/6 | 62.0 | 373.0 | 7.43E-04 | - | - |
| | 100D | 3/19/8 | 151.0 | 284.0 | 1.47E-01 | - | - |
| $Q=D$ vs. $Q=1.5D$ | 30D | 0/28/2 | 100.5 | 364.5 | 6.42E-03 | - | - |
| | 50D | 0/28/2 | 133.0 | 302.0 | 6.61E-02 | - | - |
| | 100D | 0/29/1 | 171.5 | 293.5 | 2.06E-01 | - | - |
| $Q=D^2/25$ vs. $Q=1.5D$ | 30D | 0/25/5 | 115.5 | 319.5 | 2.67E-02 | - | - |
| | 50D | 0/29/1 | 231.0 | 204.0 | 7.62E-01 | - | - |
| | 100D | 2/27/1 | 270.5 | 164.5 | 2.47E-01 | - | - |

| TABLE 17. Comparison between the one-step backtracking and random backtracking in TBT-SaDE/Mexp ($D=30$). |
|-----------------------------------|-----------------|-----------------|-----------------|-----------------|
| TBT-SaDE/Mexp | $+/=-$ | $R^+$ | $R^-$ | $p$-value | $\alpha=0.05$ | $\alpha=0.1$ |
| Random backtracking vs. One-step backtracking | 0/28/2 | 191.5 | 243.5 | 5.67E-01 | - | - |

| TABLE 18. Comparison between The Runtimes in DE/rand/1 and TBT-DE/rand/1. |
|-----------------------------------|-----------------|-----------------|-----------------|-----------------|
| Runtime (s) | DE/rand/1 | TBT-DE/rand/1 | Extra overhead |
| 30D | 3.79E+02 | 4.30E+02 | 13.46% |
| 50D | 7.39E+02 | 8.15E+02 | 10.28% |
| 100D | 2.15E+03 | 2.27E+03 | 5.58% |

| TABLE 19. Comparison between the runtimes in JADE and TBT-JADE. |
|-----------------------------------|-----------------|-----------------|-----------------|
| Runtime (s) | JADE | TBT-JADE | Extra overhead |
| 30D | 1.51E+02 | 2.18E+02 | 44.37% |
| 50D | 3.59E+02 | 4.72E+02 | 31.48% |
| 100D | 1.45E+03 | 1.74E+03 | 20.00% |
I. COMPARISON ON RUNTIME

In this section, we compare the runtime of DE/rand/1 and TBT-DE/rand/1, and the runtime of JADE and TBT-JADE. The average runtime for all 30 CEC2014 test problems are provided in TABLE 18 and TABLE 19. As seen from TABLE 18 and TABLE 19, TBT-DEs need more runtime than the corresponding DEs. But it should be highlighted that the extra overhead (TBT-DE - DE) decreases as the dimension increases, which accords with the above computational complexity analysis results.

VI. CONCLUSIONS

Traditional DE suffers from stagnation and premature convergence. To deal with these issues, this paper proposes the tracking mechanism (TM) and backtracking mechanism (BTM), respectively. The tracking mechanism is able to make the population converge to a promising area by tracking the evolution path of the current superior individuals when the population falls into the stagnation status. Backtracking mechanism can make the population escape from the premature convergence status by reallocating the premature individual to its previous positions. The experimental results show that both TM and BTM can improve the performance of DE methods (including basic DE methods and some state-of-the-art DE methods), and the contribution can be accumulated by combining TM and BTM. Moreover, our proposed framework TBT is also better than two other general frameworks, i.e. SPS framework and GAR framework.

In future, it is still an interesting research topic about how to adjust the backtracking step size for breaking away premature convergence. Moreover, whether TBT is useful or not for other population-based algorithms will be worthy of study.

REFERENCES


GENGHUI LI received the M.Sc. degree from Shenzhen University, Shenzhen, China, in 2016. He is currently pursuing the Ph.D. degree in computer science with the City University of Hong Kong, Hong Kong. His research interests include evolutionary computation, computational intelligence, and machine learning.

SHU YANG received the B.S. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2009, and the Ph.D. degree in computer science and technology from Tsinghua University, Beijing, in 2014. He is currently an Associate Research Fellow with the College of Computer Science and Software Engineering, Shenzhen University, China. His research interests include future Internet architecture, edge computing, IoT, and blockchain.

ZHONG MING received the Ph.D. degree in computer science from Sun Yat-sen University, Guangzhou, China. He is currently a Professor with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. He has authored over 100 high-quality papers in top conferences and journals, such as the IEEE TRANSACTIONS ON COMPUTERS, the ACM Transactions on Embedded Computing Systems, the ACM Transactions on Autonomous and Adaptive Systems, and the IEEE International Conference on Data Mining. His current research interests include cloud computing, internet of things, software engineering, and artificial intelligence.

ZHENKUN WEN received the M.E. degree in computer science and engineering from Tsinghua University, Beijing, China, in 1999. He is currently a Professor with the College of Computer Science and Software Engineering, Shenzhen University, Guangdong, China. His research interests include computer networks, machine learning, video information security, and so on.

NAN LU received the Ph.D. degree from Jilin University, Changchun, China. He is currently a Professor with the College of Computer Science and Engineering, Shenzhen University, China. He led several projects of Guangdong Province and Shenzhen City. His research interests include commerce intelligence, machine learning, complex network community structure, and trust mining of social network.

JIAN LU received the Ph.D. degree in applied mathematics from Shanghai Jiao Tong University, Shanghai, China, in 2007. He is currently a Professor with the College of Mathematics and Statistics, Shenzhen University, Shenzhen, China. His current research interests include image processing and intelligent computing.

***