Optimal trajectory searching based differential evolution

Dazhi Jiang, Liyu Li and Jian Gong

Department of Computer Science,
Shantou University,
GuangDong, China
Email: jiangdazhi111007@sina.com
Email: 12lyli2@stu.edu.cn
Email: 13jgong2@stu.edu.cn

Zhun Fan*

Department of Electronic and Information Engineering,
Shantou University,
GuangDong, China
Email: zfan@stu.edu.cn
*Corresponding author

Abstract: A well-designed best solution selection scheme is usually beneficial to enhance the effectiveness and efficiency of evolutionary algorithms. Optimal trajectory searching, a new best solution selection mechanism in the search behaviour of the traditional differential evolution (DE), is presented and analysed. Then an improved algorithm based on traditional DE and optimal trajectory searching mechanism is presented for enhancing the performance of DE. Performance compared with other four DE variants indicates that our algorithm outperforms them in terms of convergence speed and solution accuracy.

Keywords: optimal trajectory searching; differential evolution; search behaviour.

Reference to this paper should be made as follows: Jiang, D., Li, L., Gong, J. and Fan, Z. (2015) 'Optimal trajectory searching based differential evolution', *Int. J. Wireless and Mobile Computing*, Vol. 8, No. 4, pp.384–393.

Biographical notes: Dazhi Jiang received his BA in Computer Science from the China University of Geoscience (Wuhan) in 2004. He obtained his PhD from the State Key Laboratory of Software Engineering, Wuhan University, China in 2009. Since then, he has been with the Department of Computer, Shantou University, China, where he is an Associate Professor. His research interests include evolutionary algorithm, automatic programming, data mining and applications of artificial intelligence.

Liyu Li obtained his BS in Electronic Information Science and Technology from Hubei University of Technology in 2012 and MS in Computer Technology Application from Shantou University in 2015, respectively. His research interest includes evolutionary computation, deep learning and artificial intelligence.

Jian Gong obtained his BS in Electronic Information Engineering from Hubei University of Technology in 2013. Since then, he has been with the Department of Computer, Shantou University, China, where he is a graduate student. His research interest includes evolutionary computation and data mining

Zhun Fan is a Professor of Electronic and Information Engineering at the University of Shantou, Guangdong, China. She received her PhD in Electrical and Computer Engineering from Michigan State University. Her current research focuses on evolutionary algorithm, intelligent mechatronic systems, mechatronic design automation, etc.

1 Introduction

Inspired by the natural evolution of species to solve optimisation problems, Differential Evolution, which was written as DE for short, is arguably one of the most competitive evolutionary algorithms for continuous optimisation (Storn and Price, 1997; Moraglio and Togelious, 2009; Chen et al., 2013). Compared with the traditional evolutionary algorithms (Rana and Zaveri, 2013), DE is also a reliable and effective global optimiser algorithm that has been successfully applied in a wide range of application (Wang et al., 2013). The behaviour of DE has attracted much attention in the research community during the past decade. It was influenced not only by mutation and crossover operators, but also by the adaptive parameters controlling and search strategies in DE algorithm (Zaharie, 2007; Zielinski et al., 2006; Gaemperle et al., 2002).

In terms of adaptive parameters controlling, feedback from the evolutionary search is used to dynamically change the control parameters (Wang et al., 2013). For factor F adaptation in DE, Ali and Torn (2004) introduced auxiliary population and automatic calculating of the amplification factor F. To develop a self-adaptive control mechanism to adjust the parameters F and C_r, Brest et al. (2006) introduced self-adapting control parameter settings into DE (SADE). Qin and Suganthan (2005) emphasised the adaptation of parameter settings during the evolving procedure and accordingly established a self-adaptive DE called SaDE for numerical optimisation (Huang et al., 2006; Qin et al., 2009). Yong et al. established a neighbourhood search strategy for DE (NSDE), which uses Gaussian and Cauchy distributed functions to generate parameter F (Yang et al., 2008a). Although SaDE and NSDE have quite different emphases on improving DE's performance, Yang et al. (2008b) proposed SaNSDE (self-adaptive NSDE) which introduces SaDE's self-adaptive mechanisms into NSDE.

As to developing new search strategies in DE algorithm, Noman and Iba (2008) adopted an adaptive hill climbing strategy that uses a crossover-based adaptive local search operation to enhance the DE algorithm performance. Following a learning strategy to calculate the opposite solutions of current population, Rahnamayan et al. (2008) developed a novel opposition-based DE (ODE) algorithm to accelerate the convergence speed in differential evolution. Some research showed that the performance of DE can be improved by combining several effective trail vector generation strategies with some suitable control parameter settings (Mallipeddi et al., 2011; Wang et al., 2011).

However, most of DE variants mentioned above are concreted on rand/1, rand-to-best/1 and rand/2 schemes of DE. There has been no method developed solely based on greedy DE variants (such as DE/best/1 and DE/best/2) that utilises the information of the best solution(s) in the current population. The reason seems straightforward: a greedy variant may lead to premature convergence (Zhang and Sanderson, 2009). However, we note that in the processes of problem solving, a well-designed best solution selection scheme is usually beneficial to enhance the effectiveness and efficiency of algorithm.

In view of the above consideration, we developed an improved DE algorithm that adapts an optimal trajectory searching strategy in mutation, which could be considered as a variant of DE/best/1 algorithm (DE/best-local/1). As a generalisation of DE/best/1, DE/best-local/1 utilises the best solution information in local populations. This idea comes after the research of geometric characteristic of DE According to the geometric characteristic in DE, we found the new candidates are generated by mutation around the fixed trajectories. When searching candidates in some fixed trajectories (optimal trajectories), the fitness of those candidates always outperform candidates generated with the trajectories remained. In spite of its greedy property, the proposed strategy just makes effort in several individuals which are selected from population so that the problems such as premature convergence can be alleviated. The study shows that, an optimal trajectory searching mechanism included in the search behaviour of mutation improves the efficiency and effectiveness of problem solving.

The paper proceeds as follows. Section 2 briefly introduces the traditional DE algorithms. In Section 3, we explain the concept of geometric interpretation in DE, especially the optimal trajectory searching in DE. Section 4 presents DE algorithms based on optimal trajectory search. The experimental verifications of our DE algorithm are illustrated in Section 5. Finally, the concluding section draws some general implications and points to what remain.

2 A brief introduction to differential evolution

DE is a population-based, direct, robust and efficient search method. Like other evolutionary algorithms, DE starts with an initial population vector randomly generated in the solution space (Storn and Price, 1997). Assume N is a constant number which presents the size of population, and D is the dimension of parameter vectors. So, the population is expressed as $X_i(t)$, where $i = 1, 2, \ldots, N$, t is the generation. The main difference between DE and other evolutionary algorithms, such as Genetic Algorithm and Particle Swarm Optimisation algorithm, is its new generation vectors generating method. In order to generate a new population vectors, three vectors in population are randomly selected, and weighted difference of two of them is added to the third one. After crossover, the new vector is compared with the predetermined vector in population. If the new vector is better than predetermined one, replace it; else, the predetermined vector saved in the next generation's population. For traditional DE, the procedure is illustrated as following:

Mutation: For each vector *i* from generation *t*, a mutant vector $X_i(t+1)$ is defined by

$$X_i(t+1) = X_{r_1}(t) + F(X_{r_2}(t) - X_{r_3}(t)) \tag{1}$$

where $i \in \{1,2,\cdots,N\}$ and $r_1, r_2, r_3 \in [0,N]$, i, r_1, r_2 and r_3 are different. The differential mutation parameter *F*, known as scale factor, is a positive real normally between 0 and 1, but it also can take values greater than 1. Simply, larger values for *F* result in higher diversity in the generated population and the lower values in faster convergence.

Furthermore, mutant vector $X_i(t+1)$ could be defined by other mutation strategies, such as:

$$X_i(t+1) = X_{best}(t) + F(X_{r_2}(t) - X_{r_3}(t)) \tag{2}$$

$$X_i(t+1) = X_i(t) + F(X_{best}(t) - X_i(t)) + F(X_{r_1}(t) - X_{r_2}(t)) \tag{3}$$

$$X_i(t+1) = X_{best}(t) + F(X_{r_1}(t) - X_{r_2}(t)) + F(X_{r_3}(t) - X_{r_4}(t)) \tag{4}$$

$$X_i(t+1) = X_{r_1}(t) + F(X_{r_2}(t) - X_{r_3}(t)) + F(X_{r_4}(t) - X_{r_5}(t)) \tag{5}$$

where $X_{best}(t)$ is the best vector in generation *t*, and *i*, r_1, r_2, r_3, r_4 and r_5 are different numbers which generated from $[0,N]$. Schemes (1) and (3), which were notated as DE/rand/1/bin and DE/current to best/2/bin, are used frequently in literature due to the high performance.

Crossover: Crossover also plays an important role in DE algorithm which increases the diversity of the population. A crossover vector $X_i'(t+1)$ is defined as following:

$$X_i'(t+1) = (X_{i,j}'(t+1), X_{i,j}'(t+1), \ldots, X_{i,j}'(t+1))$$

where $j \in \{1,2,\cdots,D\}$ and

$$X_{i,j}'(t+1) = \begin{cases} X_{i,j}(t+1), & if\ rand(j) \le C_r, \\ X_{i,j}(t), & else. \end{cases}$$

The recombination probability parameter C_r takes values in $[0,1]$, and $rand(j) \in [0,1]$.

3 Geometric analyses of DE

Using (1), denote $X_i(t+1)$ by *U* with $X_{r_1}(t)$, $X_{r_2}(t)$ and $X_{r_3}(t)$ by X_1, X_2 and X_3, respectively. Thus, (1) can be rewritten as follows:

$$U + F \cdot X_3 = X_1 + F \cdot X_2 \tag{6}$$

Divide both sides by 1+*F* and let $\omega = \frac{1}{1+F}$, then we will have:

$$\omega \cdot U + (1-\omega) \cdot X_3 = \omega X_1 + (1-\omega) \cdot X_2 \tag{7}$$

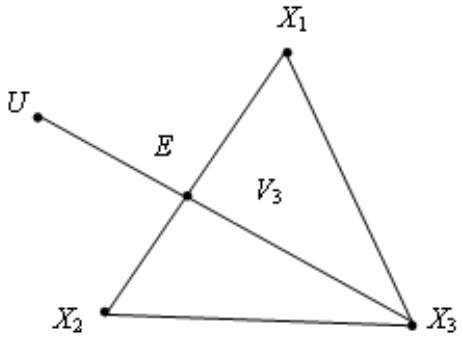
In (7), with $F \in [0,1]$, $\omega \in [0.5,1]$. Both sides of (7) are a typical form of arithmetic crossover operators in traditional evolutionary algorithms.

Supposing $E = \omega X_1 + (1-\omega) \cdot X_2$, get $\omega \cdot U + (1-\omega) \cdot X_3 = E$, and *U* can be rewritten as follows.

$$U = \frac{E - (1-\omega)X_3}{\omega} = \frac{1}{\omega}E + \left(1 - \frac{1}{\omega}\right)X_3 \tag{8}$$

The algebraic operations on real vectors in (8) can be represented graphically as in Figure 1.

Figure 1 Construction of *U* with parent vectors



Fortunately, a geometric description of (7) in terms of convex combinations (see Figure 1) can be allowed for interpretation. In Figure 1, calling *E* the vector obtained by the convex combinations with both sides of (7). Geometrically the point *E* must be the intersection points of the segments X_1X_2 and $UX3$. For the new point generated by (8), it can be determined geometrically by firstly determine *E* as convex combination of X_1 and X_2; then, a new point *U* obtained by projecting X_3 beyond *E* (extension ray).

Definition: *Suppose that V_3 is the convex space combined by X_1, X_2 and X_3, thus the point U will always be out of the V_3.*

Proof: According to Figure 1, to prove *U* is not in the V_3, if and only if *U* does not belong to the segment *EX*3. With simplified equation $U = \frac{1}{\omega}E + \left(1 - \frac{1}{\omega}\right)X3$. For $\omega \in [0.5,1]$, hence $\frac{1}{\omega} \in [1,2]$ and $1 - \frac{1}{\omega} \in [-1,0]$. With the parameters $\frac{1}{\omega}$ and $1 - \frac{1}{\omega}$, the new point *U* obtained by the linear combination of points *E* and X_3 is out of the segment *EX*3, which means that *U* is not between *E* and X_3.

Supposing *U* is in the segment *EX*3, if and only if $\frac{1}{\omega}, 1 - \frac{1}{\omega} \in [0,1]$ and $\frac{1}{\omega} + \left(1 - \frac{1}{\omega}\right) = 1$.

The process of mutation for traditional DE can be interpreted as follows (use scheme 1). Selecting three vectors X_1, X_2 and X_3 in $N(t)$, for any ω^*, where

$\omega^* \in [0.5,1]$, intersection points E^* can be obtained by the formula $\omega^* X_1 + (1-\omega^*) \cdot X_2$. Then, according to the ω^* and E^*, new points U^* will be obtained by formula of $\frac{1}{\omega^*} E^* + \left(1 - \frac{1}{\omega^*}\right) X3$.

For this process, considering two cases of ω^* as follows.

1 $\omega^* = 1$

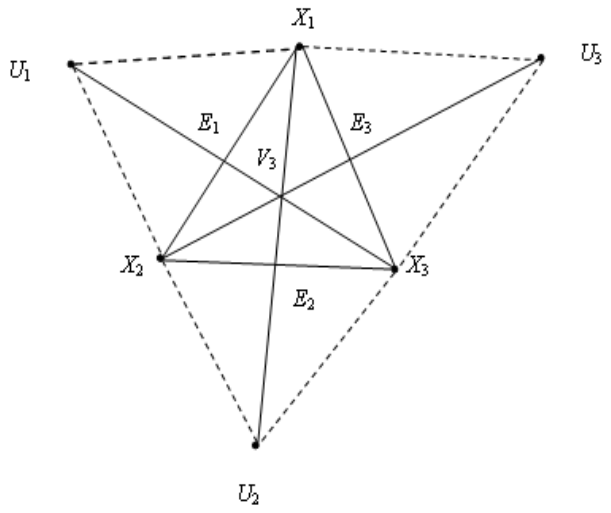
With two equations $E^* = \omega^* X_1 + (1-\omega^*) \cdot X_2$ and $U^* = \frac{E^* - (1-\omega^*) X_3}{\omega^*}$, it is obvious that points X_1, E^* and U^* are all concentrated together where X_1 located before mutation.

2 $\omega^* = 0.5$

Similarly, with equation $E^* = \omega^* X_1 + (1-\omega^*) \cdot X_2$, point E^* is the midpoint of segment $X_1 X_2$. While, with equation $U^* = \frac{E^* - (1-\omega^*) X_3}{\omega^*}$, point U^* is located in the extension ray of segment $X_3 E^*$, and E^* is the midpoint of segment $X_3 U^*$ simultaneously.

With these two cases, we can construct a more detailed geometric expression for DE mutation (Figure 2).

Figure 2 Geometric expression for DE mutation



The process of mutation for DE can be interpreted in detail as follows. Selecting three vectors X_1, X_2 and X_3 in $N(t)$, with $\omega^* \in [0.5,1]$, get

$$U^* \in \{X_1U_1, X_1U_3, X_2U_1, X_2U_2, X_3U_2, X_3U_3\}.$$

Beside, $\{X_1U_1, X_1U_3, X_2U_1, X_2U_2, X_3U_2, X_3U_3\} \cap V_3 = \{X_1, X_2, X_3\}$.

4 DE algorithm based on optimal trajectory searching

4.1 Optimal trajectory in DE algorithm

For the DE/rand/1/bin algorithm, according to the geometric expression of DE, new point U^* is determined by E^* (determined by X_1, X_2) and X_3. It is obviously that X_1, X_2 and X_3 have distinguishing function in the mutation equation $U = X_1 + F \cdot (X_2 - X_3)$.

Here we consider X_1, X_2 and X_3 as three different locations. Thus for three individuals X_i, X_j and X_k which is randomly selected in population, there are six kinds of combination values when the individuals are located in the different locations. According to the fitness of X_i, X_j and X_k, there are three cases for location X_1, which are the individual with best fitness, the medium fitness and the worst one. In the view of fitness discrepancy, $U = X_1 + F \cdot (X_2 - X_3)$ could be transformed into three new forms as follows.

Case 1: best individual in location X_1:

$$U = X_{best} + F \cdot (X_{middle} - X_{worst})$$

$$U = X_{best} + F \cdot (X_{worst} - X_{middle})$$

Case 2: middle individual in location X_1:

$$U = X_{middle} + F \cdot (X_{best} - X_{worst})$$

$$U = X_{middle} + F \cdot (X_{worst} - X_{best})$$

Case 3: worst individual in location X_1:

$$U = X_{worst} + F \cdot (X_{best} - X_{middle})$$

$$U = X_{worst} + F \cdot (X_{middle} - X_{best})$$

When X_i, X_j and X_k are located in the determined location, the trajectory for generated new candidates is determined in the meanwhile. After that, the algorithm will search candidates in the fixed searching trajectories. For the *Case 1*, the trajectories are X_1U_1 and X_1U_3; for the *Case 2*, the trajectories are X_2U_1 and X_2U_2; for the *Case 3* are X_3U_2 and X_3U_3.The question is which are the optimal searching trajectories when generating new candidates with mutation operator?

Through experimental observation that the offspring generated by *Case 1* have the highest probability to surpass the father individuals compared with DE, *Case 2* and *Case 3*. It means that the optimal trajectories in DE mutation $U = X_1 + F \cdot (X_2 - X_3)$ are X_1U_1 and X_1U_3.

4.2 The framework of DEOTS

The evolving framework of algorithm is constructed as follows:

Evolving Framework of DEOTS:

Initialise population of N_p vectors at random, $t = 0$

While stop criterion not met do

For all vectors $X_i(t)$ in the population do

Mutation strategy:

Step I: Pick at random three distinct vectors from the current population $X_{r_1}(t)$, $X_{r_2}(t)$ and $X_{r_3}(t)$, where, $r_1 \neq r_2 \neq r_3$.

Step II: Find the vector $X_r(t)$ with best fitness in $X_{r_1}(t)$, $X_{r_2}(t)$ and $X_{r_3}(t)$, where $r \in \{r_1, r_2, r_3\}$.

Step III: Create intermediate vector $X_i(t+1)$ with $X_i(t+1) = X_r(t) + F(X_a(t) - X_b(t))$, where $a, b \in \{r_1, r_2, r_3\} \cap \{r\}$ and $a \neq b$.

Crossover strategy:

Create vector $X_i'(t+1)$

with $X'_{i,j}(t+1) = \begin{cases} X_{i,j}(t+1), & if\ rand(j) \le C_r, \\ X_{i,j}(t), & else. \end{cases}$, where

C_r is the recombination parameter.

Selection strategy:

If $f(X_i'(t+1))$ better than $f(X_i(t))$ then

Set the *i*-th vector in the next population

$Y_i(t) = X_i'(t+1)$

Else

$Y_i(t) = X_i(t)$

End if

End for

For all vectors $X_i(t)$ in the population do

Set $X_i(t+1) = Y_i(t)$, $t = t+1$

End for

End while

5 Experimental verification

All test functions applied in this experiment are well-known benchmark functions which have been frequently used in literature (Yao et al., 1999). All of these functions used in this paper are minimisation problems. The conducted experiments are categorised in two groups in order to investigate the performance of DEOTS. To evaluate the performance of convergence accurately, the number of function calls (NFC) was employed.

5.1 Experiment 1

In Experiment1, DEOTS is compared with some excellent algorithms, which are SaDE, NSDE and SaNSDE. The following parameters are used by DEOTS in current experiment:

- Population size, *N*=100.
- Dimension, *D*=30.
- *F*=*Rand*(0.0,1.0).
- $C_r = 0.5$.
- Maximum number of function calls (MAX_{NFC}): 1500*100 for **F1-F4**, 5000*100 for **F5**, 1500*100 for **F6-F13**, 200*100 for **F14**, 1500*100 for **F15** and 200*100 for **F16-F21**.

The average results of 20 independent trials on functions **F1** to **F21** are summarised in Table 1. Especially, the results for three algorithms, SaDE, NSDE and SaNSDE, are cited from Yang et al. (2008b).

Table 1 Comparison of DEOTS with SaDE, NSDE and SaNSDE, where Mean Best indicates the average best fitness values

F	*MAXNFC*	*SaDE Mean Best*	*NSDE Mean Best*	*SaNSDE Mean Best*	*DEOTS Mean Best*
F1	150000	7.49E-20	7.76E-16	3.02E-23	**1.23E-41**
F2	150000	6.22E-11	4.51E-10	4.64E-11	**8.01E-25**
F3	150000	1.12E-18	1.06E-14	6.62E-22	**2.04E-39**
F4	150000	2.96E-02	2.54E-02	1.59E-03	**1.32E-243**
F5	500000	2.10E+01	1.24E+01	**4.13E-30**	4.36 E+00
F6	150000	**0.0**	**0.0**	**0.0**	**0.0**
F7	150000	7.58E-03	1.20E-02	7.21E-03	**4.78E-03**
F8	150000	**–12569.5**	**–12569.5**	**–12569.5**	–12557.6
F9	150000	**4.00E-08**	7.97E-02	1.84E-05	3.58
F10	150000	9.06E-11	6.72E-09	2.36E-12	**5.95E-15**
F11	150000	8.88E-18	6.72E-09	**0.0**	**0.0**
F12	200000	1.21E-19	5.63E-17	5.94E-23	**1.57E-32**
F13	200000	1.75E-19	5.52E-16	3.12E-22	**1.35E-32**
F14	200000	**0.998**	**0.998**	**0.998**	**0.998**
F15	200000	**3.07E-4**	**3.07E-4**	**3.07E-4**	**3.07E-4**
F16	200000	**–1.03**	**–1.03**	**–1.03**	**–1.03**
F17	200000	**0.398**	**0.398**	**0.398**	**0.398**
F18	200000	**3**	**3**	**3**	**3**
F19	200000	**–10.15**	**–10.15**	**–10.15**	**–10.15**
F20	200000	**–10.40**	**–10.40**	**–10.40**	**–10.40**
F21	200000	**–10.54**	**–10.54**	**–10.54**	**–10.54**

For functions F1-F4 and F7, DEOTS achieved better than SaDE, NSDE and SaNSDE, typically in F1-F4. For function F10, DEOTS, as well as SaNSDE, obtain the same performance. For functions F6 and F8, all four algorithms performed exactly the same. SaNSDE achieved the best in function F5, and the DEOTS better than SaDE and NSDE clearly. The most difference from the results is the function F9, where SaDE performs best and, DEOTS performs the most trivial among them.

Figure 3 Convergence curves of DE and DEOTS for test functionsF1–F9. X axis represents number of function calls and Y axis represents the best fitness. (a) **F1**. (b) **F2**. (c) **F3**. (d) **F4**. (e) **F5**. (f) **F6**. (g) **F7**. (h) **F8**. (i) **F9** (see online version for colours)

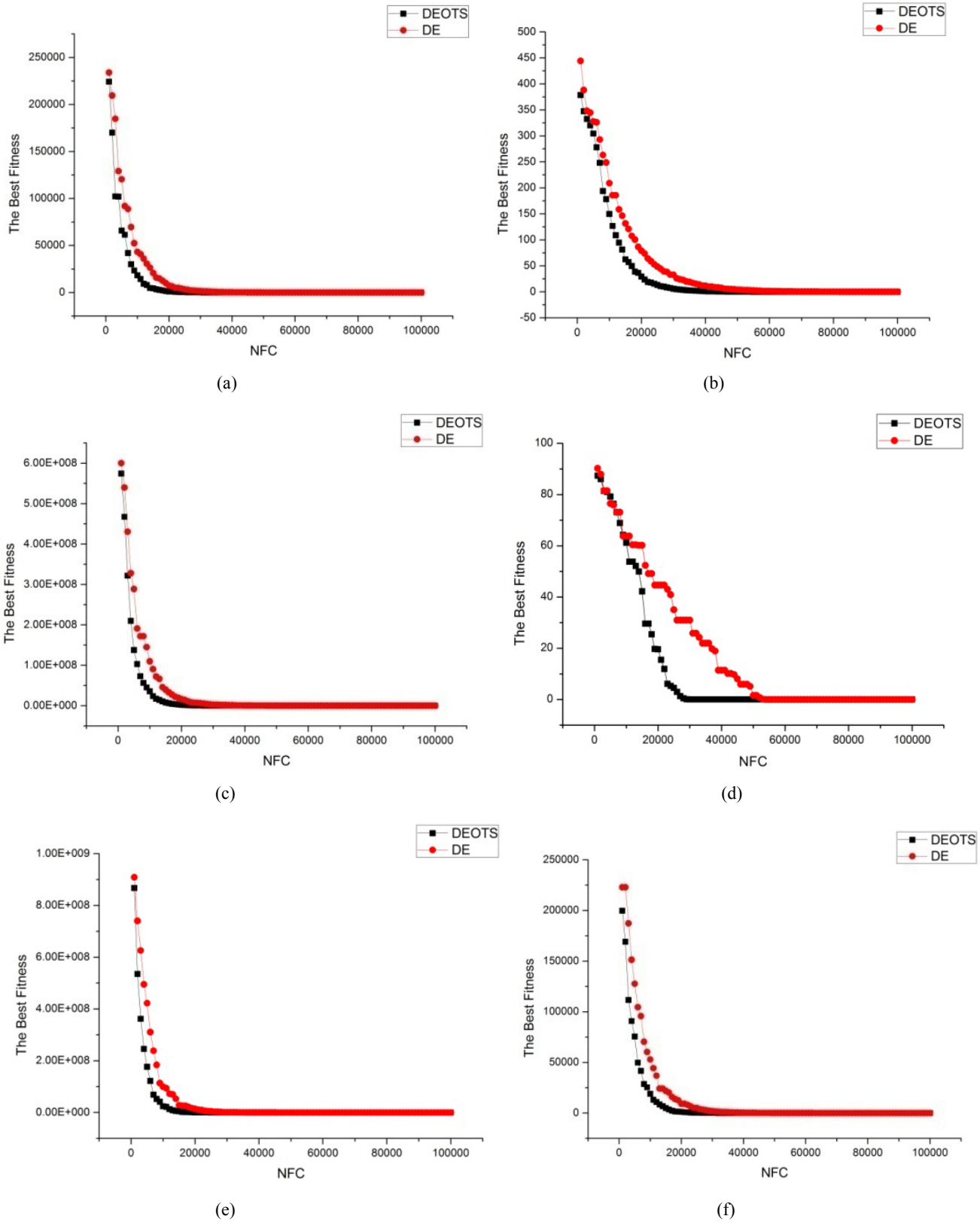
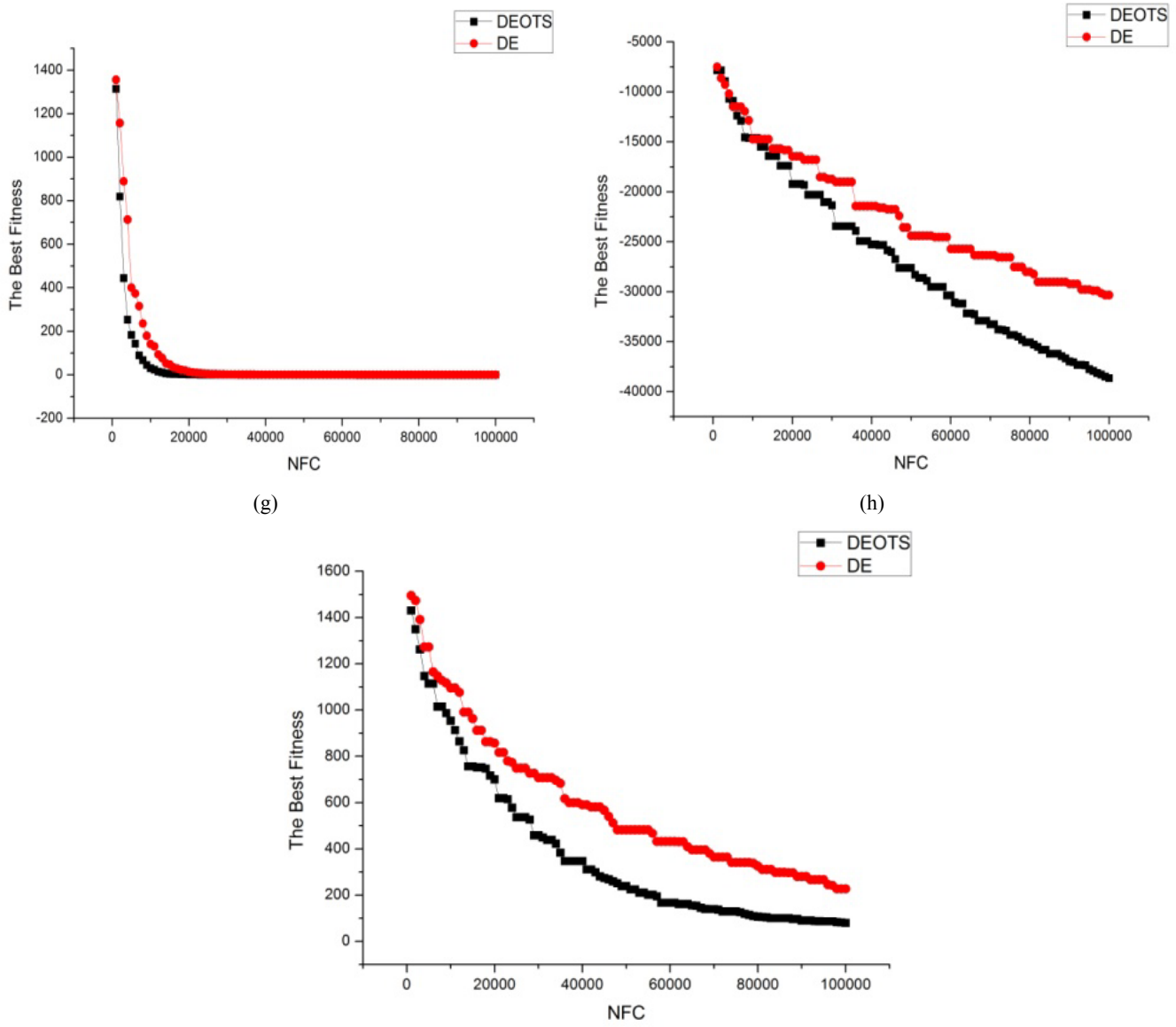


Figure 3 Convergence curves of DE and DEOTS for test functionsF1–F9. X axis represents number of function calls and Y axis represents the best fitness. (a) **F1**. (b) **F2**. (c) **F3**. (d) **F4**. (e) **F5**. (f) **F6**. (g) **F7**. (h) **F8**. (i) **F9** (see online version for colours) (continued)



(g)

(h)

(i)

The convergence comparisons between DE and DEOTS are shown in Figure 3. For simplicity, each figure shows the result of a random trial. Because of space limitation, just some samples are presented. Actually, DEOTS converges to a better optimal solution than DE in the anaphase of evolution (sometimes when number of function calls is bigger than 20000 approximately) in every trail. What's more, with the help of optimal trajectory searching, the DEOTS converges quickly compared with DE.

The significant improvement achieved by DEOTS can be attributed to the local optimisation around the optimal individuals. With the optimal trajectory searching, algorithm generated new competitive candidates around the X_1. When the X_1 has the best fitness, the mutation operator will generate the new candidates in better positions with a high probability. Therefore, DEOTS performs better in terms of convergence speed and solution accuracy.

5.2 Experiment 2

In SaDE, two candidate strategies are used to improve the performance of algorithm (Qin and Suganthan, 2005). For this purpose, two probabilities are applied to decide which strategy is chosen for each individual in current population. Moreover, for adaptability, the probabilities are changed in the evolution according to the ratio of trial vectors successfully entering the next generation and the vectors discard. A strategy with bigger success times, to some extent, indicates that it has more potential to generate good candidates.

Thus, for Experiment 2, a hybrid algorithm is constructed with two mutation strategy DE/rand/1/bin and DE/best-local/1 which focuses on the times the trail vectors successfully enter the next generation. In hybrid algorithm, two mutation strategies generated trail vectors respectively, then compared for each individual in the current population. Assuming the success times of trail vectors entering the next generation by applying DE/rand/1/bin and DE/best-local/1 are recorded as $\boldsymbol{ts}_1$ and $\boldsymbol{ts}_2$ respectively. Those two numbers are accumulated within a specified number of function calls (1000 in our experiments). After the accumulated period, both of them will be reset for another accumulation step.

All experiments in Experiment 2 have been repeated 20 times with different random initialisation, for each benchmark function to obtain statistically reliable performance numbers. For time saving, only eight test functions (F1-F9) are selected to test for Experiment 2.

Parameters Settings: The parameter settings for comparison success times between hybrid algorithms are as follows.

- Test functions: F1~F8.
- Population size, N=100.
- Dimension, D=100.
- F=*Rand*(0.0,1.0)
- C_r = 0.5.
- Maximum number of evaluations (MAX_{NFC}):1E+06 for all test functions.

As seen in the Figure 4, st2 is frequently bigger than the st1 obviously (especially in Figure 4 (a, b, c, d, e, f, g)), and it means that mutation strategy with DE/best-local/1 has more chance and potential to generate good candidates compared with traditional DE.

Figure 4 The success times of trail vectors entering the next generation by applying DE mutation (ts1) and DEOTS mutation (ts2) for test functionsF1-F9. X axis represents a period of number of function calls and Y axis represents the success times. (a)**F1**. (b) **F2**. (c) **F3**. (d) **F4**. (e) **F5**. (f) **F6**. (g) **F7**. (h) **F8**. (i) **F9** (see online version for colours)

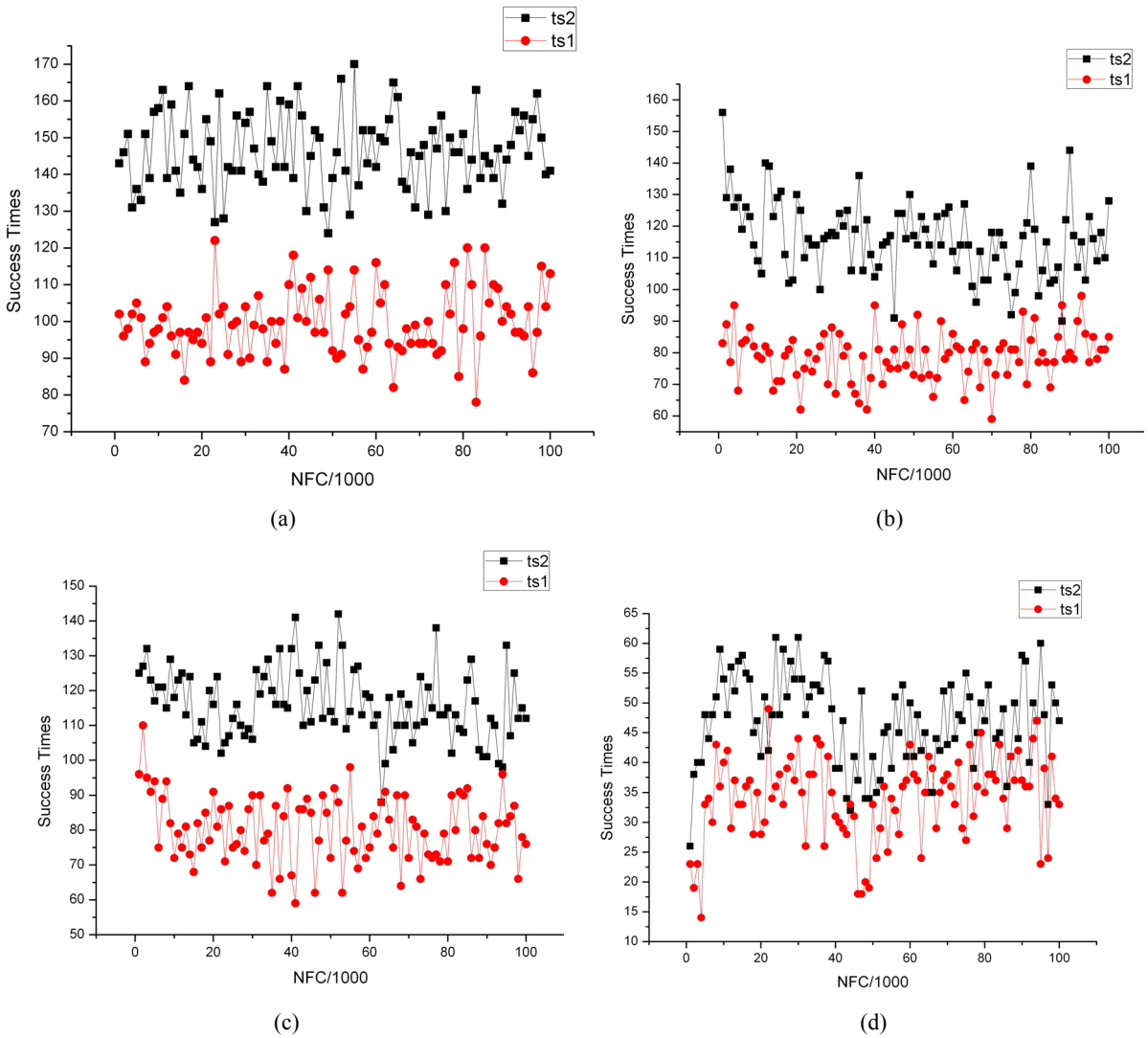
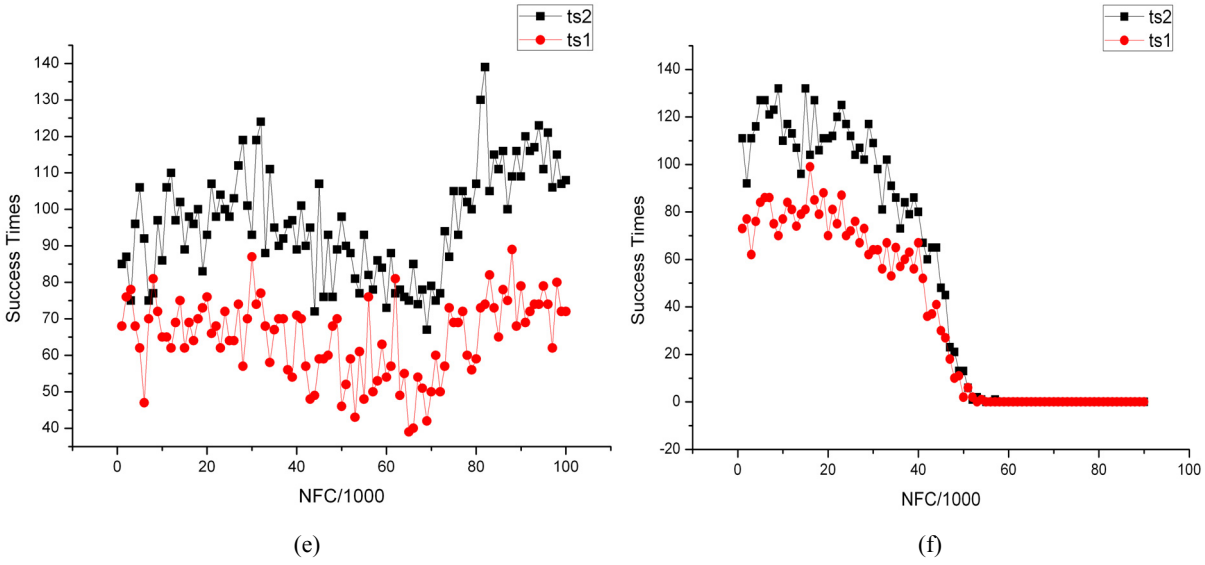
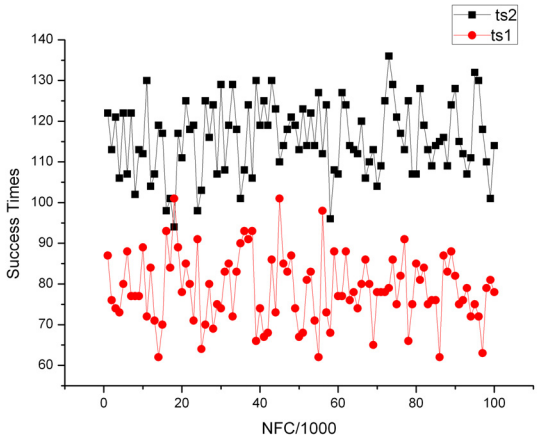


Figure 4 The success times of trail vectors entering the next generation by applying DE mutation (ts1) and DEOTS mutation (ts2) for test functionsF1-F9. X axis represents a period of number of function calls and Y axis represents the success times. (a)**F1**. (b) **F2**. (c) **F3**. (d) **F4**. (e) **F5**. (f) **F6**. (g) **F7**. (h) **F8**. (i) **F9** (see online version for colours) (continued)

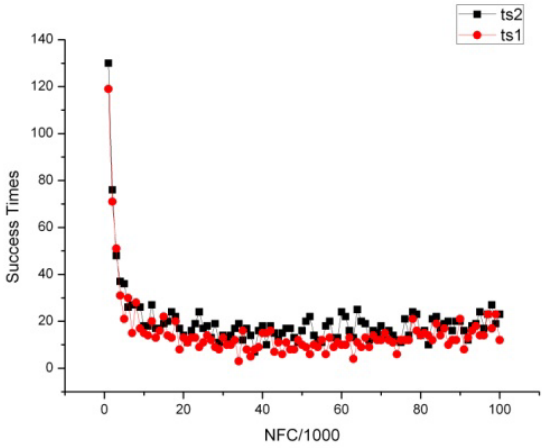


(e)

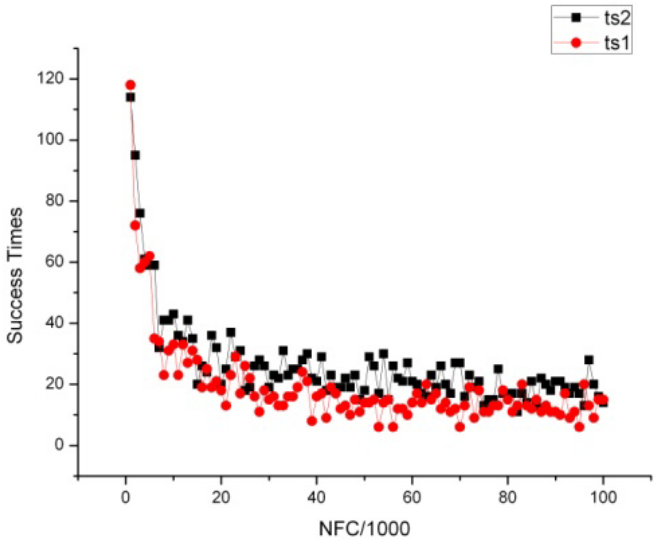
(f)



(g)



(h)



(i)

6 Conclusion and future work

Reference to the DE's common mutation strategies, such as DE/rand/1/bin, DEOTS can be donated as DE/rand/1/bin/ Arrange algorithm which the vectors chose for mutation is sorted with a given ordering before mutation.

In this paper, optimal trajectory is analysed based on the geometric characteristic of DE. With optimal trajectory searching mechanism, an effective algorithm named as DEOTS is presented in this paper. According to the analyses of ejection mechanism and numerical experiments, generating new candidates around the optimal trajectory is more likely to get the global optimal solutions (for minimum problems).

The DEOTS achieves better performance in many well-known benchmark functions. As an improved DE algorithm, DEOTS does not introduce any additional parameters into the DE algorithm to enhance the efficiency compared with SaDE, NSDE and SaNSDE. Actually, the number and the setting of parameters for DEOTS are same as the traditional DE. In DEOTS, such tiny arrangement which can make so much significant effort is exciting and delightful for algorithm designer, while useful and helpful for algorithm designing.

Optimal trajectory searching has a high potential to improve performance of optimisation algorithm and can be embedded with various mutation strategies of DE. This work presents preliminary results of optimal trajectory searching. The future works will focus on the combinations of optimal trajectory searching and other DE algorithms, such as DE/ best/1/bin, DE/best/2/bin, DE/rand/2/bin, etc. Moreover, some related works on parameter or strategy adaptation in evolutionary algorithms have been done in many literatures. Another aspect of future research is the adaptations of the learning strategies and parameters settings during the evolution procedure.

Acknowledgements

The authors would like to thank the anonymous reviewers for their very detailed and helpful comments that help us to increase the quality of this work. This work was supported by Natural Science Foundation of Guangdong Province (No.: S2013010013974), in part by the Shantou University National Foundation Cultivation Project (No.: NFC13003), in part by the National Natural Science Foundation of China (No.: 61175073), in part by the Leading Talent Project of Guangdong Province.

References

Ali, M.M. and Torn, A. (2004) 'Population set-based global optimization algorithms: some modifications and numerical studies', *Computers & Operations Research*, Vol. 31, No. 10, pp.1703–1725.

Brest, J., Greiner, S., Boskovic, B., Mernik, M. and Zumer, V. (2006) 'Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems', *IEEE Transactions on Evolutionary Computation*, Vol. 10, No. 6, pp.646–657.

Chen, L., Wang, W. and Wang, H. (2013) 'Accelerating Gaussian bare-bones differential evolution using neighbourhood mutation', *International Journal of Computing Science and Mathematics*, Vol. 4, no. 3, pp.266–276.

Gaemperle, R., Mueller, S.D. and Koumoutsakos, P. (2002) 'A parameter study for differential evolution', in Grmela, A., Mastorakis, N.E. (Eds): *Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, WSEAS Press, pp.293–298.

Huang, V.L., Qin, A.K. and Suganthan, P.N. (2006) 'Self-adaptive differential evolution algorithm for constrained real-parameter optimization', *Proceedings of the IEEE Congress on Evolutionary Computation* (*CEC'06*), Vancouver, BC, Canada, July.

Mallipeddi, R., Suganthan, P.N., Pan, Q.K. and Tasgetiren, M.F. (2011) 'Differential evolution algorithm with ensemble of parameters and mutation strategies', *Applied Soft Computing*, Vol. 11, No. 2, pp.1679–1696.

Moraglio, A. and Togelious, J. (2009) 'Geometric differential evolution', *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, pp.1705–1712.

Noman, N. and Iba, H. (2008) 'Accelerating differential evolution using an adaptive local search', *IEEE Transactions on Evolutionary Computation*, Vol. 12, No. 1, pp.107–125.

Qin, A.K. and Suganthan, P.N. (2005) 'Self-adaptive differential evolution algorithm for numerical optimization', *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, Edinburgh, UK, September.

Qin, A.K., Huang, V.L. and Suganthan, P.N. (2009) 'Differential evolution algorithm with strategy adaptation for global numerical optimization', *IEEE Transactions on Evolutionary Computation*, Vol. 13, No. 2, pp.398–417.

Rahnamayan, S., Tizhoosh, H.R. and Salama, M.M.A. (2008) 'Opposition-based differential evolution', *IEEE Transactions on Evolutionary Computation*, Vol. 12, No. 1, pp.64–79.

Rana, K. and Zaveri, M. (2013) 'Energy-efficient routing for wireless sensor network using genetic algorithm and particle swarm optimization techniques', *International Journal of Wireless and Mobile Computing*, Vol. 6, No. 4, pp.392–406.

Storn, R. and Price, K. (1997) 'Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces', *Journal of Global Optimization*, Vol. 11, pp.341–359.

Wang, H., Lu, Y. and Peng, W. (2013) 'Permutation flow-shop scheduling using a hybrid differential evolution algorithm', *International Journal of Computing Science and Mathematics*, Vol. 4, No. 3, pp.298–307.

Wang, Y., Cai, Z. and Zhang, Q. (2011) 'Differential evolution with composite trial vector generation strategies and control parameters', *IEEE Transactions on Evolutionary Computation*, Vol. 15, No. 1, pp.55–66.

Yang, Z., He, J. and Yao, X. (2008a) 'Making a difference to differential evolution', *Advance in Metaheuristics for Hand Optimization*, pp.397–414.

Yang, Z., Tang, K. and Yao, X. (2008b) 'Self-adaptive differential evolution with neighborhood search', *IEEE Congress on Evolutionary Computation*, pp.1110–1116.

Yao, X., Liu, Y. and Lin, G. (1999) 'Evolutionary programming made faster', *IEEE Transactions on Evolutionary Computation*, Vol. 3, No. 2, pp.82–102.

Zaharie, D. (2007) 'A comparative analysis of crossover variants in differential evolution', *Proceedings of the International Multi Conference on Computer Science and Information Technology*, pp.171–181.

Zhang, J. and Sanderson, A.C. (2009) 'JADE: adaptive differential evolution with optional external archive', *IEEE Transactions on Evolutionary Computation*, Vol. 13, No. 5, pp.945–958.

Zielinski, K., Weitkemper, P., Laur, R. and Kammeyer, K-D. (2006) 'Parameter study for differential evolution using a power allocation problem including interference cancellation', *IEEE Congress on Evolutionary Computation*, Vancouver, BC, Canada.