Dynamic differential evolution with oppositional orthogonal crossover for large scale optimisation problems

Xiaogang Dong, Changshou Deng* and Yucheng Tan

School of Information Science and Technology,
JiuJiang University,
JiuJiang, China
Email: dxg110@jju.edu.cn
Email: dengtju@aliyun.com
Email: yctan@jju.edu.cn
*Corresponding author

Abstract: Differential evolution is a population-based optimisation algorithm and has been successfully applied in many fields. However, when tackling large-scale optimisation problem, it still encounters serious challenges. To meet these challenges, a dynamic differential evolution with oppositional orthogonal crossover is proposed in this paper. A new oppositional learning is proposed and then it is used in the orthogonal crossover to improve the exploitation ability of the dynamic differential evolution. During the evolution process in dynamic differential evolution, only one individual is randomly chosen to undergo this oppositional orthogonal crossover operation. Thirteen benchmark problems with 1,000 dimensions were used to evaluate its performance. The results show that the proposed method is very competitive in terms of solution quality obtained.

Keywords: large-scale optimisation; dynamic differential evolution; DDE; oppositional orthogonal crossover; OOC; new oppositional learning.

Reference to this paper should be made as follows: Dong, X., Deng, C. and Tan, Y. (2017) 'Dynamic differential evolution with oppositional orthogonal crossover for large scale optimisation problems', *Int. J. Computing Science and Mathematics*, Vol. 8, No. 5, pp.414–424.

Biographical notes: Xiaogang Dong received his Master in Computation Mathematics from the Huazhong University of Science and Technology. He is a Lecturer at the Jiujiang University, Jiujiang, Jiangxi, PRC. His current research focuses on swarm intelligence.

Changshou Deng received his PhD from the University of Tianjin in 2007. He is currently a Professor at the Jiujiang University, Jiujiang, Jiangxi, PRC. His current research interest includes nature-inspired computation and application.

Yucheng Tan received his Master in Computation Mathematics from the Jiangxi Normal University. He is an Associate Professor at the Jiujiang University, Jiujiang, Jiangxi, PRC. His current research interest is evolutionary computation.

This paper is a revised and expanded version of a paper entitled 'Dynamic differential evolution with oppositional orthogonal crossover for large scale optimization problems' presented at 7th International workshop on swarm intelligent systems and 5th International Workshop on Mobile and Wireless Computing, Hang Zhou, 28 October 2016.

1 Introduction

Differential evolution (DE) (Storn and Price, 1997) is a simple yet powerful evolutionary optimisation method in continuous search domain. DE has been successfully applied for solving many real world problems in science and engineering (Wang et al, 2013; Chen et al, 2014; Zhang, 2015). Like other evolutionary algorithms (Cai et al, 2016; Cui et al, 2017), although DE has shown excellent performance in tackling small or medium sized problems, it still encounters serious challenges when applying to large scale problems. Firstly, the problem scale appears to be the main factor that contributes to the complexity of an optimisation problem. Secondly, the search space of a problem grows exponentially with the number of variables and the properties of the search space may alter as the number of dimension increases. Thirdly, the evaluation of large-scale problems is usually expensive. There are two solutions to overcome these difficulties. One solution is based on decomposition and the other one is to use hybridisation approach. Although the decomposition approach is beneficial to overcome dimensionality problem, DE still needs to be more reliable and powerful when it is applied to non-separable large scale problem (Sayed et al, 2012).

Brest et al. (2006) presented a self-adaptive DE, in which the control parameters F and CR are associated with each individual and then can be adapted automatically with other two new parameters. Dynamic DE (DDE) proposed by Qing (2006) updates the population dynamically. In DDE, the evolution of each individual may improve other individuals immediately without the need to wait until the next generation. The trial vectors are always generated using the newly updated population. Thus it can explore the larger space during evolution. Wang et al. (2011) proposed a composite DE which utilises three trial vector generation strategies and three groups of parameter settings. Rahnamayan et al. (2008) enhanced the performance of DE with the concept of opposition-based learning (OBL). Wang et al. (2013) proposed GPU-based generalised OBLDE for large scale optimisation problems. Yong et al. (2012) used an orthogonal crossover to enhance the search ability of DE (OXDE). Yang et al. (2008) proposed competitive DECC-G for large scale optimisation problems, particular non-separable problems. In this article, based on OBL and orthogonal learning, a novel oppositional orthogonal crossover (OOC) is introduced and it is hybridised with DDE for large scale optimisation.

The remainder of this paper is organised as follows. Section 2 introduces DE. DDE with OOC is presented in Section 3. Experiments are carried out in Section 4 and lastly Section 5 concludes this paper.

2 Differential evolution

DE is a population-based optimisation algorithm and maintains a population of NP D-dimensional vectors. DE improves its population with mutation, crossover and selection operations generation by generation until the termination condition has been met.

After being initialised, DE employs mutation operation to generate a mutation vector v of each target vector x according to equation (1).

$$v_{i,G+1} = x_{r1,G} + F\left(x_{r2,G} - x_{r3,G}\right) \tag{1}$$

where i, $r1$, $r2$, $r3 \in \{1, 2, \ldots, NP\}$ are randomly chosen and have to be mutually exclusive, F is the scaling factor for the difference between the individual x_2 and x_3, G is generation.

After the mutation operation, DE employs the crossover operation to generate a trial vector which is the mixture of the target vector and the mutation vector. The traditional DE employs the binomial crossover defined as equation (2):

$$u_{i,G+1} = \begin{cases} v_{i,g+1}, & \text{if } rand < CR \text{ or } j = rand(i) \\ x_{i,G}, & otherwise \end{cases} \tag{2}$$

where $i = 1, 2, \ldots, NP$, $j = 1, 2, \ldots, D$, $CR \in [0, 1]$ is the crossover probability and $rand(i) \in \{1, 2, \ldots, D\}$ is the randomly selected number which ensures that the trial vector ($u_{i,G+1}$) gets at least one element from the mutation vector ($v_{i,G}$) and D is the scale of the problem.

Finally, the target vector $x_{i,G}$ is compared with the trial vector in terms of the objective value to determine which one will survive in the next generation:

$$x_{i,G+1} = \begin{cases} u_{i,g+1}, & \text{if } f\left(u_{i,G+1}\right) < f\left(x_{i,G}\right) \\ x_{i,G}, & otherwise \end{cases} \tag{3}$$

3 DDE with OOC

As aforementioned, the experimental design method and OBL have been used to enhance the search ability of DE. DDE updates the population dynamically and responds to any improvement immediately. Thus it can explore the larger space during evolution. To effectively balance the exploration and exploitation ability of DDE, OOC was introduced. OOC is a novel operation which combines both advantages of orthogonal experiment and OBL. Then, DDE combing with OOC is applied to solve large scale optimisation problem.

3.1 Orthogonal crossover

Experimental design techniques can be applied to enhance crossover operation (Yong et al, 2012). As a matter of fact, a crossover is a procedure for sampling several points from a defined region while orthogonal crossover operator (OC) is to generate a small but representative of sampling points. Leung and Wang (2001) used OC with quantisation

technique (QOC) and proposed new OC used for dealing with numerical optimisation for continuous variables.

QOC is based on a predefined orthogonal array. An orthogonal array for K factors with Q levels and M combinations is often recorded as $L_M(Q^K)$.For example, an experiment with four factors and three levels, its orthogonal array is $L_9(3^4)$ (Yong et al, 2012).There are 81 combinations for all. If the orthogonal array is used, then only nine combinations are considered. The orthogonality of an orthogonal array means that each level of the factor occurs the same times for any column and each possible combination of levels of the factors occurs the same times as well for any two columns.

In QOC, the genes in p_1 and p_2 are quantified into Q levels such that the difference between any two successive levels is the same. The i^{th} level, denoted by $Level(i) = (l_{i,1}, l_{i,2}, \ldots, l_{i,k})$ where $i = 1, 2, \ldots, Q$ and $l_{i,j}$ is defined as follows:

$$l_{i,j}\begin{cases} \min\left(p_{1,j}, p_{2,j}\right), & \text{for } i = 1 \text{ and } 1 \le j \le k \\ \min\left(p_{1,j}, p_{2,j}\right) + (i-1) * \left(\dfrac{\left|p_{1,j} - p_{2,j}\right|}{Q-1}\right), & \text{for } 2i \le Q-1 \text{ and } 1 \le j \le K \\ \max\left(p_{1,j}, p_{2,j}\right), & \text{for } i = Q \text{ and } 1 \le j \le k \end{cases} \tag{4}$$

For large scale optimisation problems, the D is usually much larger than K. For instance, D = 1,000 and K = 4 as shown in the following experiment. We group the different variables into K sub-vectors with the way in Leung et al. (2001).Finally, M individuals are generated according to orthogonal array $L_M(Q^K)$ and the best individual is chosen. In this article, we use $L_9(3^4)$ to define an orthogonal crossover and it is named OC9, thus one individual survives from nine combined individuals. OC9 can be regarded as a mapping from (D, D) to D, i.e.:

$$z = OC9(x, y) \tag{5}$$

The OC9 operator was presented as Algorithm 1.

Algorithm 1 OC9

Input:	Two vectors x, y
1	Divide the large-scale vectors into four subgroups
2	Construct three levels according to equation (4)
3	Generate nine trial vectors by combing elements from vectors x, y according to $L_9(3^4)$
4	Evaluate the objective function values of the nine trial vectors.
Output:	The vector which has the smallest value among the nine trial vectors.

3.2 New oppositional learning

OBL originates from reinforcement learning and has been successfully applied in several soft computing methods. To accelerate the convergence speeds of evolutionary algorithms, Tizhoosh et al. (2006) is the first to extend genetic algorithm by taking the opposite of each chromosome and thus creating a corresponding opposition-chromosome. The definitions of opposition are as follows.

Definition 1: let $x \in [a, b]$ be any real number. Its opposite, *ox* is defined as

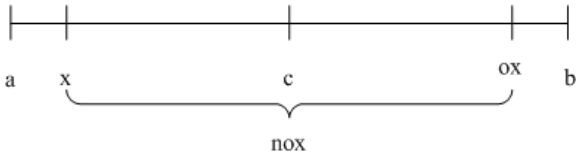
$$ox = a + b - x \tag{6}$$

Definition 2: let $x \in [a, b]$ be any real number. Then its new oppositional point, *nox* is defined as

$$nox = rand(x, ox) \tag{7}$$

where *ox* is the opposite of *x*.

Figure 1 Opposite points defined in domain filed [a, b]. c is the centre of the domain and x is a de individual. ox is the opposite of x and new oppositional point nox, respectively



If *x* is an individual in DE, then *ox* is the opposite of *x*, *nox* is the new oppositional point, respectively. Figure 1 illustrates a point *x*, its opposition, *ox* and the new opposition, *nox*, as defined in equations (6–7).

3.3 DDEOOC

During the course of evolution in DDEOOC, only one individual is chosen to undergo the OOC operation and the rest of individuals undergo the same crossover in DDE. New mutation is used for the individual who is selected to undergo the OOC operation.

$$v_i = x_{r1} + rand(0, 1)(x_{r2} - x_{r3}) \tag{8}$$

where *rand*(0, 1) is a number which is uniformly generated between zero and one. The detail of DDEOOC is as Algorithm 2.

Algorithm 2 DDEOOC

Input: NP; F; CR; MaxFEs (maximum number of function evaluations); orthogonal array

1 Randomly initialise population P with NP individual
2 Evaluate the objective function value f for each individuals
3 FEs = NP
4 while FEs ≤ MaxFEs do
5 for i = 1: NP do
6 kk = rand[1, NP]
7 if (kk equals to i) then goto 8 else goto 14
8 undergo mutation operation according to equation (8)
9 undergo OC9 operator to generate nine trial individuals
10 to generate nine oppositional individuals for the above nine trial individuals according to equation (7)

```
11    evaluate the 18 individuals and select the best one to exist
12    Fes = Fes + 18;
13    goto 15
14    undergo mutation operation and crossover operation of DDE
15    undergo selection operation of DDE
16    Fes = Fes + 1;
17    end for
18    end while
```

Output: the best individual with smallest fitness

4 Experimental study

4.1 Experimental setup

To investigate the performance of DDEOCC for large scale optimisation, the well-studied benchmark problems (Yao et al., 1999) are chosen as a test bed. The algorithms used for comparison include three other competitive DE variants (jDE, OXDE, CoDE) and one well-known cooperative co-evolution DE (DECC-G). The number of decision variables, was set to 1,000 for all the 13 test functions. For each method and each test problem, 25 independent runs were conducted with 5E6 as maximum number of function evaluations (MaxFEs). The population size was set to 100, the scaling factor and crossover probability were both set to 0.9. The parameters in original paper of jDE, OXDE, CoDE and DECC-G were used.

4.2 Results on benchmark functions

The results of all functions are shown in Table 1, where 'mean' indicates the mean function value and 'std' represents the corresponding standard. The results of the best algorithms are marked in boldface.

The statistical comparison of the mean value of DDEOOC with other algorithms using Wilcoxon's rank sum test was conducted on the experimental results. In table 1, '–' indicates that the performance of DDEOOC is better than that of compared algorithms (DDEOOC wins), '≈' indicates that there is no significant difference between the compared two algorithms (DDEOOC ties), '+' indicates that the performance of DDEOOC is worse than that of the compared algorithm (DDEOOC loses). The last row of Table 1 summarises the experimental results with win/lose/tie(w/l/t).

The results in Table 1 clearly show that the proposed DDEOOC outperforms the other three DE variants (OXDE, jDE and CoDE) on all the thirteen problems. DECC-G is the best one on problems F5 and F8. On problem F6, both DDEOOC and DECC-G can find the optimum and there is no significant difference between them in term of performance. DDEOOC surpasses the competitive DECC-G on the rest ten problems.

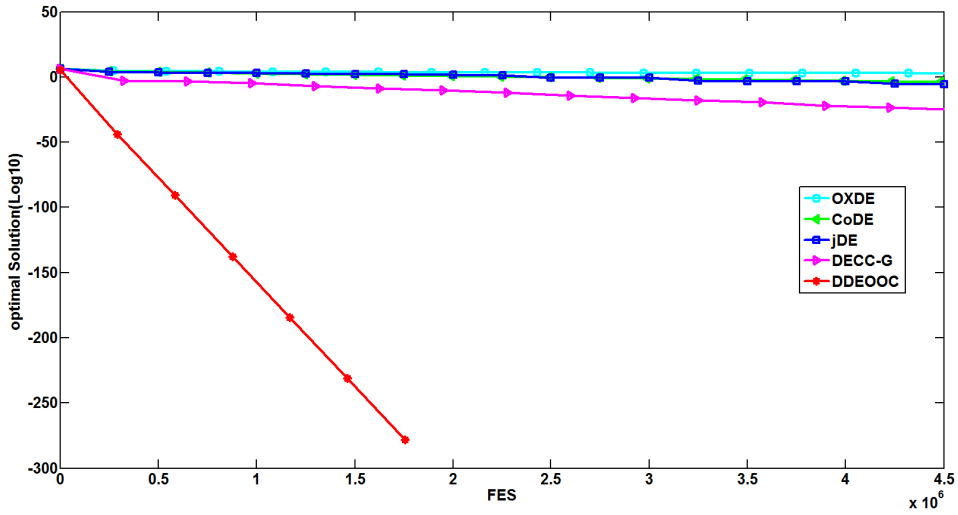
Table 1 Experimental Results of OXDE, jDE, CoDE, DECC-G and DDEOOC over 25 independent runs on 13 test functions of 1,000 variables with 5,000,000 FES

Fun	*OXDE(mean ± std)*	*jDE(mean ± std)*	*CoDE(mean± std)*	*DECC-G(mean ± std)*	*DDEOOC(mean ± std)*
F1	4.76e+02 ± 1.67e+02-	2.46e-06 ± 1.23e-05-	1.50e-05 ± 1.74e-05-	9.61e-29 ± 3.11e-29-	0.00e+00 ± 0.00e+00
F2	5.27e+01 ± 6.89e+00-	1.74e-10 ± 8.62e-10-	8.89e-01 ± 9.74e-01-	1.70e-14 ± 1.77e-14-	0.00e+00 ± 0.00e+00
F3	1.54e+06 ± 2.33e+05-	7.54e+04 ± 1.48e+04-	4.50e+04 ± 6.12e+03-	1.20e+03 ± 6.70e-04-	0.00e+00 ± 0.00e+00
F4	2.59e+01 ± 1.84e+00-	5.04e+01 ± 4.58e+00-	2.67e+01 ± 1.95e+00-	3.19e-02 ± 4.72e-03-	1.11e-315 ± 0.00e+00
F5	2.28e+04 ± 7.08e+03-	2.18e+03 ± 2.21e+02-	2.39e+03 ± 2.12e+02-	9.86e+02 ± 4.11e-01+	9.89e+02 ± 8.30e-03
F6	7.86e+03 ± 8.86e+02-	1.06e+04 ± 2.98e+03-	5.26e+01 ± 6.26e+01-	0.00e+00 ± 0.00e+00-	0.00e+00 ± 0.00e+00
F7	5.68e+00 ± 7.25e-01-	2.91e+01 ± 1.60e+01-	9.06e-01 ± 1.03e-01-	2.62e-03 ± 6.68e-04-	2.69e-05 ± 1.65e-05
F8	–4.17e+05 ± 7.27e+002-	–4.19e+05 ± 3.18e-01-	–1.89e+05 ± 5.14+03-	–4.19e+05 ± 9.28e-11+	–4.19e+05 ± 1.19e-10
F9	4.94e+02 ± 5.32e+01-	2.98e+00 ± 4.03e+00-	4.59e+03 ± 2.10e+02-	1.25e-14 ± 7.49e-15-	0.00e+00 ± 0.00e+00
F10	6.49e+00 ± 2.79e-01-	5.12e+00 ± 7.06e-01-	2.25e+00 ± 1.82e-01-	1.27e-13 ± 7.11e-15-	8.89e-16 ± 0.00e+00
F11	5.02e+00 ± 1.19e+00-	8.91e-01 ± 7.39e-01-	7.70e-03 ± 2.29e-02-	9.50e-16 ± 1.44e-16-	0.00e+00 ± 0.00e+00
F12	3.01e+00 ± 7.11e-11-	1.32e+06 ± 2.20+06-	5.75e-02 ± 4.85e-02-	1.24e-27 ± 3.30e-28-	4.71e-34 ± 1.75e-49
F13	1.94e+03 ± 3.40e+02-	1.14e+07±8.17e+06-	1.15e+02 ±7.53e+01-	2.64e-03 ± 4.79e-03-	1.35e-032 ± 5.59e-48
w-l-t	13-0-0	13-0-0	13-0-0	10-2-1	

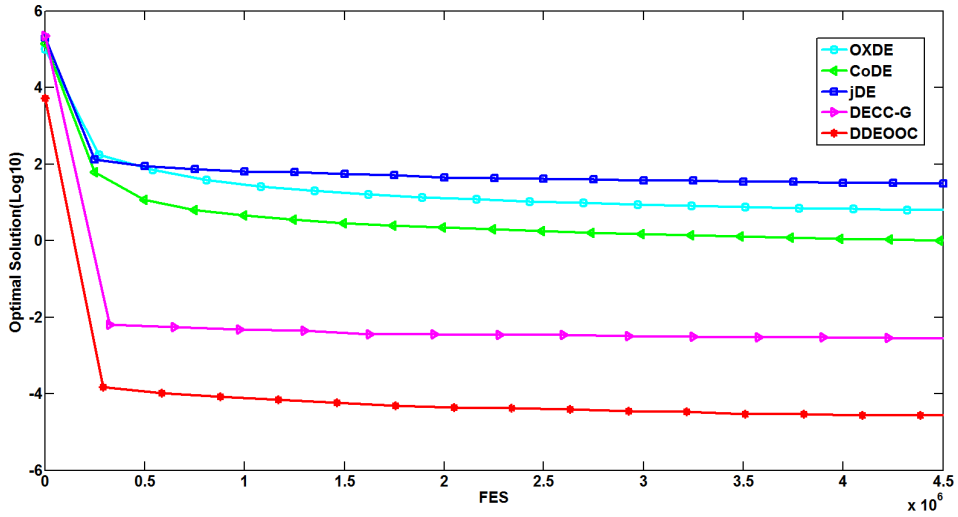
Table 2 Average rankings achieved by Friedman test

Algorithms	*DDEOOC*	*DECC-G*	*CoDE*	*jDE*	*OXDE*
Test value	1.19	1.88	3.62	3.92	4.38

Figure 2 Convergence curves, (a) f_1 (b) f_7 (c) f_{10} (d) f_{12} (see online version for colours)

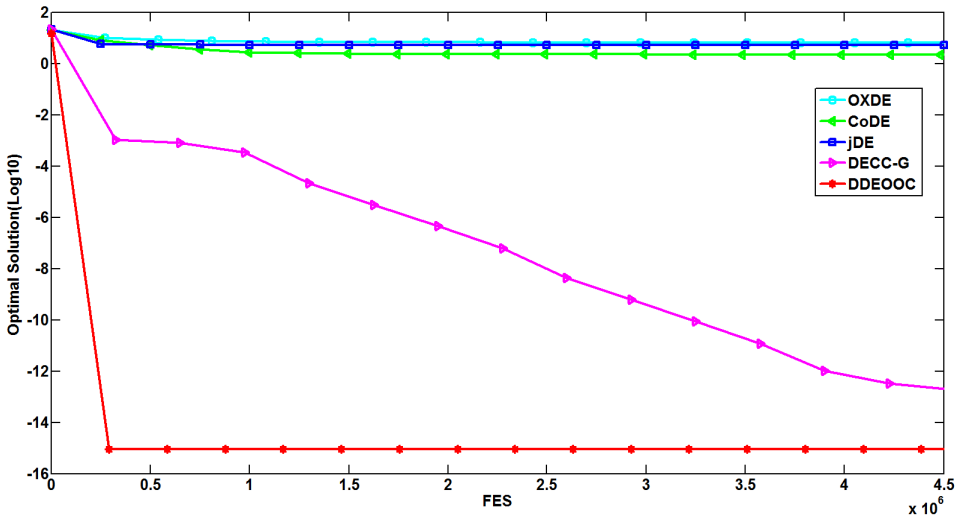


(a)

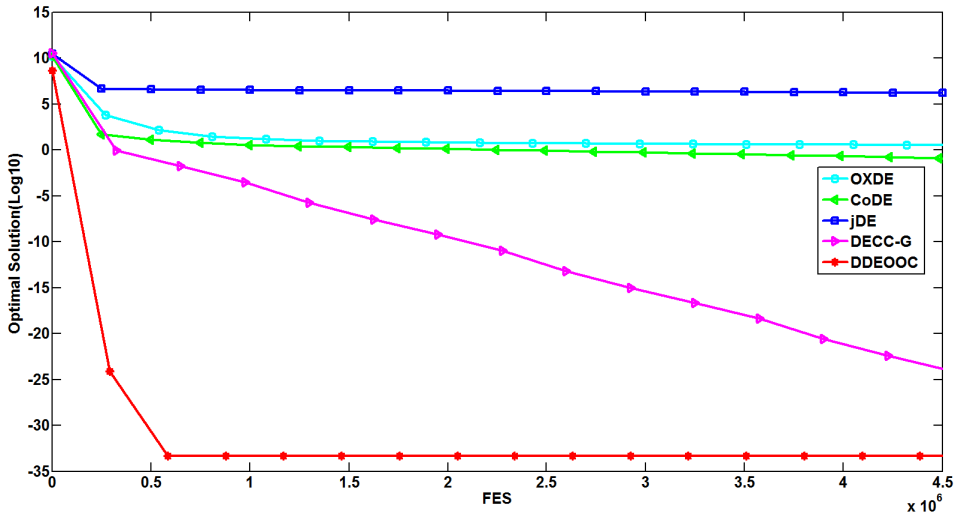


(b)

Figure 2 Convergence curves, (a) f_1 (b) f_7 (c) f_{10} (d) f_{12} (continued) (see online version for colours)



(c)



(d)

Figure 2 presented the convergence curves of the five methods on four selected problems. The Friedman test was used to compare the performance of five methods on the test suite. Table 2 shows the average rankings of the five DE variants where the best ranking is highlighted in boldface. The best average ranking is obtained by the DDEOOC, which outperforms the other four algorithms. The performance of the rest of algorithms on the 13 problems can be sorted by average ranking into the following order: DDEOOC, DECC-G, CoDE, jDE and OXDE.

Results in Tables 1–2 and Figure 2 show clearly that DDEOOC is a competitive method for large scale optimisation.

5 Conclusions

DE still suffers from the curse of dimensionality' when solving large scale optimisation problem. To overcome this problem, DDEOOC was proposed. In DDEOOC, we introduced a novel opposition based learning strategy and then applied it in the orthogonal crossover to enhance the exploitation ability of DDE. During the evolution of DDEOOC, only one individual is randomly chosen to conduct this new crossover operator, thus it does not need much additional computing cost. The performance of the proposed method was evaluated with 13 benchmark problems. The comparison with four other well-known DE variants shows that the proposed method can serve as a competitive algorithm for large scale optimisation problem. In the future, it is very interesting to investigate whether the OOC can be used to enhance other population-based algorithm performance, such as particle swarm optimisation algorithm and artificial bee colony algorithm for large scale optimisation problem.

Acknowledgements

This work is partially supported by Natural Science Foundation of China under grant No. 61364025.

References

Brest, J. et al. (2006) 'Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems', *IEEE Transactions on Evolutionary Computation*, December, Vol. 10, No. 6, pp.646–657.

Cai, X.J. et al. (2016) 'Improved bat algorithm with optimal forage strategy and random disturbance strategy', *International Journal of Bio-Inspired Computation*, Vol. 8, No. 4, pp.205–214.

Chen, W. and Ni, X. (2014) 'Chaotic differential evolution algorithm for resource constrained project scheduling problem', *International Journal of Computing Science and Mathematics*, Vol. 5, No. 1, pp.81–93.

Cui, Z.H. et al. (2017) 'A novel oriented cuckoo search algorithm to improve DV-hop performance for cyber-physical systems', *Journal of Parallel and Distributed Computing*, Vol. 103, No. C, pp.42–52.

Leung, Y.W. et al. (2001) 'An orthogonal genetic algorithm with quantization for global numerical optimization', *IEEE Transactions on Evolutionary Computation*, February, Vol. 5, No. 1, pp.41–53.

Qing, A. (2006) 'Dynamic differential evolution strategy and applications in electromagnetic inverse scattering problems', *IEEE Transactions on Geoscience and Remote Sensing*, December, Vol. 44, No. 1, pp.62–73.

Rahnamayan, S. and Wang, G.G. et al. (2008) 'Solving large scale optimization problems by opposition-based differential evolution', *WSEAS Transactions on Computers*, May, Vol. 7, No. 10, pp.1792–1804.

Sayed, E., Essam, D. and Sarker, R. (2012) 'Dependency identification technique for large scale optimization problems', *IEEE Congress on Evolutionary Computation*, IEEE, pp.1–8.

Storn, R. and Price, K.V.(1997) 'Differential evolution CA simple and efficient heuristic for global optimization over continuous spaces', *J. Glob. Optim.*, Vol. 11, No. 4, pp.341–359.

Tizhoosh, H.R. et al. (2006) 'Opposition-based reinforcement learning', *JACIII*, Vol. 10, No. 4, pp.578–585.

Wang, H. et al. (2013) 'Parallel differential evolution with self-adapting control parameters and generalized opposition-based learning for solving highdimensional optimization problems', *J. Parallel Distrib. Comput.*, January, Vol. 73, No. 1, pp.62–73.

Wang, H.Y., Lu, Y.B., Peng, W.L. (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. et al. (2011) 'Differential evolution with composite trial vector generation strategies and control parameters', *IEEE Transactions on Evolutionary Computation*, January, Vol. 15, No. 1, pp.55–66.

Yang, Z. et al. (2008) 'Large scale evolutionary optimization using cooperative coevolution', *Information Sciences*, Vol. 178, No. 15, pp.2985–2999.

Yao, X. et al. (2002) 'Evolutionary programming made faster', *IEEE Transactions on Evolutionary Computation*, Vol. 3, No. 2, pp.82–102.

Yong, W. et al. (2012) 'Enhancing the search ability of differential evolution through orthogonal crossover', *Inform. Sci.*, February, Vol. 185, No. 1, pp.153–177.

Zhang, Z. (2015) 'A new multi-population-based differential evolution', *International Journal of Computing Science and Mathematics*, Vol. 6, No. 1, pp.3–12.