Fouad, Allouani; Zenger, Kai; Gao, Xiaozhi

A novel Flower Pollination Algorithm based on Genetic Algorithm Operators

Published in:
Proceedings of The 9th EUROSIM Congress on Modelling and Simulation (EUROSIM 2016), The 57th SIMS Conference on Simulation and Modelling (SIMS 2016)

DOI:
10.3384/ecp17142

Published: 01/01/2018

Document Version
Publisher's PDF, also known as Version of record

Please cite the original version:
Fouad, A., Zenger, K., & Gao, X. (2018). A novel Flower Pollination Algorithm based on Genetic Algorithm Operators. In E. Juuso, E. Dahlquist, & K. Leiviskä (Eds.), *Proceedings of The 9th EUROSIM Congress on Modelling and Simulation (EUROSIM 2016), The 57th SIMS Conference on Simulation and Modelling (SIMS 2016)* (pp. 1060-1066). (Linköping electronic conference proceedings; No. 142). LINKÖPING UNIVERSITY ELECTRONIC PRESS. https://doi.org/10.3384/ecp17142

A Novel Flower Pollination Algorithm based on Genetic Algorithm Operators

Allouani Fouad [1] Kai Zenger [2] Xiao-Zhi Gao [3, 4]

[1]Department of Industrial Engineering, University of Khenchela, Algeria, fouad.allouani@g.enp.edu.dz
[2]Department of Electrical Engineering and Automation, Aalto University, Aalto, Finland, kai.zenger@aalto.fi
[3]Machine Vision and Pattern Recognition Laboratory, Lappeenranta University of Technology, Lappeenranta, Finland.
[4]School of Computing, University of Eastern Finland, Kuopio, Finland, xiao.z.gao@gmail.com

Abstract

The Flower Pollination Algorithm (FPA) is a new natural bio-inspired optimization algorithm that mimics the real-life processes of the flower pollination. Thus, the latter has a quick convergence, but its population diversity and convergence precision can be limited in some applications. In order to improve its intensification (exploitation) and diversification (exploration) abilities, we have introduced a simple modification in its general structure. More precisely, we have added both Crossover and Mutation Genetic Algorithm (GA) operators respectively, just after calculating the new candidate solutions and the greedy selection operation in its basic structure. The proposed method, called FPA-GA has been tested on all the CEC2005 contest test instances. Experimental results show that FPA-GA is very competitive.

Keywords— flower pollination algorithm, crossover, mutation, genetic algorithm (GA)

1 Introduction

Swarm intelligence (SI) optimization algorithms which are inspired by simulation of various types of biological behavior existing in nature, have characteristics of simple operation, good optimization performance and strong robustness. In the last two decades, a large number of algorithms based on this aspect have been suggested, such as, ant colony optimization (ACO) (Socha and Dorigo, 2008), differential evolution (DE) (Storn and Price, 1997), particle swarm optimization(PSO) (Kennedy and Eberhart, 1995), firefly algorithm (FA) (Xinshe, 2012), glowworm swarm optimization (GSO) (Yongquan and Jiakun, 2012), monkey search (MS) (Mucherino and Seref, 2007), harmony search (HS) (Geem *et al*, 2001), cuckoo search (CS) (Yang and Deb, 2009), bat algorithm (BA) (Yang, 2010). SI optimization algorithm can solve complex optimization problems, which classical methods cannot handle efficiently. They have shown excellent performance in many ways (Blum and Li, 2008), and their fields of application are continuously growing (Yang *et al*, 2013).

Flower pollination algorithm (FPA) is a simple and effective SI optimization algorithm proposed in (Yang, 2012). It derives its inspiration from pollination process of flowering plants. From the biological evolution point of view, the objective of flower pollination is the survival of the fittest and the optimal reproduction of plant species. All these factors involved in this process interact systematically between them to achieve optimal reproduction of the flowering plants.

In reality (in the general sense), there are two different ways of pollination; Self-pollination and cross-pollination (Yang, 2012). The cross-pollination (or global pollination) means that pollination can be achieved through pollinators, which carry pollen of a flower of a different plant using Levy flights (Yang, 2012).

The second type, self-pollination (local pollination), is made by the same plant or flower without pollinators. In the latter, the carrying process of pollen is generally done with the help of environmental factors such as wind and diffusion in the water (Yang, 2012).

In this paper, a novel FPA based on both Crossover and Mutation Genetic Algorithm (GAs) operators has been proposed. The changes made allow the introduction of two major improvements: (i) enhancing the diversity of the population, and (ii) improving the intensification ability by the association of these two operators and the elite selection mechanism. Indeed, to demonstrate the efficacy of the proposed algorithm an experimental investigation was carried out using the CEC2005 test suite benchmark problems (Suganthan *et al*, 2005). In addition, the proposed method was also compared to a set of state-of-the-art algorithms including, the basic FPA, the MGOFPA (Draa, 2015), which is a recently proposed FPA variant, the Co-variance Matrix Adaptation Evolution Strategies (CMA-ES) algorithm (Hansen and Ostermeier, 2001), the Comprehensive Learning Particle Swarm Optimizer (CLPSO) (Liang *et al*, 2006), JADE (Zhang and Sanderson, 2009), jDE (Brest *et al*, 2012), CoDE (Wang *et al*, 2011) which are all a DE variants. Moreover, Wilcoxon's rank-sum statistical test was carried out at 5 % significance level to judge whether

the results of the proposed algorithm differ from those of the other algorithms in a statistically significant way (Derrac *et al*, 2011). The rest of this paper is organized as follows: Section 2 presents the fundamental principles of the standard FPA. Section 3 contains a brief description of GAs and its crossover and mutation operators. The proposed algorithm is introduced in Section 4. Experimental results are reported in Section 5. Finally, Section 6 concludes this paper.

2 The Flower Pollination Algorithm

The flower pollination algorithm (FPA) is a new population-based optimization technique inspired by the physiological process of mating in plants. More specifically, this algorithm mimics the reproduction of plants of the same kind or other, through the so-called fertilization or pollination of flowers. To better basically understand the principle of this optimization technique, we start by giving a brief description of its biological underpinnings (Yang, 2012).

2.1 Biological underpinnings of the FPA

Generally, everyone knows that the reproduction of almost plants, in its direct and simple meaning, is a result of a pollination operation. Thus, this very important biological process is typically associated with the transfer of a chemical substance called pollen, and such transfer is often linked with some creatures called pollinators such as insects, birds, bats and other animals.

In fact, some pollinators and certain flowers have co-evolved into very specialized flower-pollinator cooperation. For example, some pollination kind cannot be completed successfully without the intervention of a specific type of pollinators. In reality, there are two main forms in the pollination process; the biotic and abiotic pollination. Thus, about 90% of flowering plants belong in the first class, in which the pollen is transferred by a specific pollinator. Concerning the second class, which does not involve using other organisms and employs wind, water or gravity as pollination mediators, we find only 10% of flowering plants.

Pollinators, or sometimes-called pollen vectors, which may be of various kinds like honeybees for example, represent an essential factor in a biotic pollination form. Thus, some pollinators tend to visit exclusively one species of flower; this pollinator behavior is called flower constancy. The latter increases directly the reproduction of the same flower species by maximizing the transfer of flower pollen to the same plants. This is also advantageous for the pollinators, since they will be sure of the availability of nectar supply with a limited memory and minimum cost of learning.

Depending on the availability of pollinators, two types of pollination are considered; self-pollination and cross-pollination. The first pollination type, called also local pollination, occurs when pollen from one flower pollinates the same flower or other flowers of the same plant (Yang, 2012). Contrariwise, cross-pollination also known as global pollination, happens over long distances when pollen is delivered to a flower from a different plant through a direct or indirect intervention of pollinators following the so-called Lévy flight behavior (Pavlyukevich, 2007).

2.2 The FPA

In (Yang, 2012), Yang emulated the characteristic of the biological flower pollination process in flowering plants to develop the algorithm in question, based on four main rules listed as follows:

Rule1: The global pollination process takes place through biotic and cross-pollination, such that the movement of pollinators has the form of the levy flight (Pavlyukevich, 2007).

Rule2: Local pollination process is considered as abiotic and self-pollination.

Rule3: The flower constancy provided by pollinators is equivalent to a reproduction probability proportional to the similarity of two flowers involved in pollination process.

Rule4: The orientation of the global pollination process, towards local or global pollination is controlled by a switch probability $p \in [0,1]$ with a simple prejudice toward local pollination for reasons relating to the approximation of the algorithm to the real case.

The implementation of these rules is based on a simplistic idea said that: each plant has only one flower, and each flower produces only one pollen gamete (Yang, 2012). Thus, this argument means that it is not necessary to distinguish between a pollen gamete, a flower, a plant or a solution to a problem.

The transition to the mathematical formulation of these rules is carried out according to (Yang, 2012) as follows; first, the global pollination processes (Rule 1), and flower constancy (Rule 3) are represented using the following equation:

$$x_i^{t+1} = x_i^t + \gamma L(\lambda)(g_* - x_i^t) \quad (1)$$

where, x_i^t is the pollen i or the solution vector x_i at iteration t, x_i^{t+1} is the generated solution vector at iteration $t+1$, $g_*$is the current best solution. In addition, γ is a scaling factor used to control the step size. $L(\lambda)$ is the Lévy flights-based step size, it corresponds to the strength of the pollination. In reality, pollinators can fly over a long distance with different distance steps; this can be modeled using a Lévy distribution (Pavlyukevich, 2007) according to the following equation:

$$L \sim \frac{\lambda\Gamma(\lambda)\sin(\frac{\pi\lambda}{2})}{\pi}\frac{1}{s^{1+\lambda}} \quad (s \gg s_0 \gg 0). \qquad (2)$$

In this equation, $\Gamma(\lambda)$ is the standard gamma function, and this distribution is valid for large steps $s > 0$.

Then, the local pollination (Rule 2), and the flower constancy (Rule 3) can be represented as follows:

$$x_i^{t+1} = x_i^t + \epsilon(x_j^t - x_k^t) \qquad (3)$$

where x_j^t and x_k^t are pollen gametes obtained from different flowers of the same plant species. Thus, this random subtraction $(x_j^t - x_k^t)$ is used to imitate the flower constancy in a limited neighbourhood. The parameter ϵ is chosen arbitrarily in $[0,1]$ to approximate this selection to a local random walk (Yang, 2012).

Flower pollination processes can occur randomly at all scales, both local and global case. Hence, to emulate this bi-orientation, a switching parameter p chosen randomly in $[0,1]$ (Rule 4) can be effectively used (Yang, 2012).

The standard FPA is summarised in the following:

Algorithm 1. The flower pollination algorithm

1: Objective Function $f(x)$, $x = (x_1, \ldots, x_d)^T$
2: Initialise a population of n_f flowers in random positions
3: Find the best solution g_* in the initial population
4: Define the switch probability $p \in [0, 1]$
5: Initialise the iteration counter $t = 0$
6: **While** $t < t_{max}$ **do**
7: For $i = 1 : n_f$ (all n_f flowers in the population) do
8: If rand < p then
9: Draw a d-dimensional step vector L which obeys a Lévy distribution
10: Do global pollination via (1)
11: Else
12: Draw ϵ from a uniform distribution in [0,1]
13: Randomly chose x_j^t and x_k^t from the population
14: Do local pollination via (3)
15: End if
16: Evaluate the newly generated solution x_i^{t+1}
17: If the newly generated solution is better, replace x_i^t by x_i^{t+1}
18: Update the current best solution g_*
19: $t = t + 1$
18: End for
19: End while

3 Genetic Algorithm

Genetic algorithm (GA) is a search method that employs random choice to guide a highly exploitative search, by maintaining a balance between exploration of the feasible search domain and exploitation of ‘‘good’’ solutions, see (Holland, 1992). A simple GA is comprised of three main operators: reproduction, crossover, and mutation. Reproduction allocates more copies to solutions with better fitness values and thus imposes the survival-of-the-fittest mechanism on the candidate solutions. Crossover combines partially a set of bits and pieces of two or more parental solutions to produce new, possibly better solutions (i.e. offspring).

Many crossover techniques exist, but the key idea of the most of them is based on the following simple concept: two individuals (parents) are randomly selected and recombined with a probability equal to p_C called crossover probability. Indeed, the combination is achieved if the following condition $rand \leq p_C$ is verified, where $rand$ is random number. Otherwise, the two offspring are simply copies of their parents.

Mutation is the occasional random inversion of bit values that generates non-recursive offspring. More precisely, mutation is often the secondary operator performed with a low probability in GAs. One of the most common mutations method is the bit-flip mutation (Sastry *et al*, 2005). In this kind of mutation, each bit in a binary string is altered (from 0 to 1 or the opposite) with a certain probability p_m known as the mutation probability. In reality, mutation operator performs a random walk near to the individual.

In this paper, we integrate these two GAs operators (crossover and mutation) in the FPA structure to improve its performances. The typical crossover and mutation operation is shown in Figure.1.

Figure 1. Crossover and mutation operation.

4 The Proposed Algorithm

The key idea of the proposed algorithm FPA-GA, is including the concept of crossover and mutation operators as successive steps in the basic FPA. These two steps are included just after calculating the new candidate solutions and the greedy selection operation. Thus, the proposed FPA-GA can be described as shown in the pseudo-code of Algorithm 2 below.

Algorithm 2. FPA based on Crossover and Mutation GAs Operators

1: Objective Function $f(x), x = (x_1, \dots, x_d)^T$
2: Initialise a population of n_f flowers in random positions
3: Find the best solution g_* in the initial population
4: Define the switch probability $p \in [0, 1]$
5: Define the crossover and mutation application probability $Prob\_CrMu \in [0, 1]$
6: $Numb_Obj_Ev = nf$n
7: **While** $Numb\_Obj\_Ev < Max\_Obj\_Ev$ **do**
9: For $i = 1 : n_f$ (all n_f flowers in the population) do
10: If rand < p then
11: Draw a d-dimensional step vector L which obeys a Lévy distribution
12: Do global pollination via (1)
13: Else
14: Draw ϵ from a uniform distribution in [0,1]
15: Randomly chose x_j^t and x_k^t from the population
16: Do local pollination via (3)
17: End if
18: Evaluate the newly generated solution x_i^{t+1}
19: If the newly generated solution is better, replace x_i^t by x_i^{t+1}
20: End for
21: $Numb\_Obj\_Ev = Numb\_Obj\_Ev + n_f$
22: If $Numb\_Obj\_Ev < Max\_Obj\_Ev$ **then**
23: Break ;
24: End if
25: if $rand < Prob\_CrMu$ **then**
26: Apply *crossover operator* on the current population Pop_t to generate a new population Pop_{Cross}
27: Apply *mutation operator* on the Pop_{Cross} population to generate a new population Pop_{Muta}
28: Apply an elite selection of n_f individuals from $\boldsymbol{Pop_t \cup Pop_{Cross} \cup Pop_{Muta}}$
29: $Numb\_Obj\_Ev = Numb\_Obj\_Ev + n_f$
30: end if
31: If $Numb\_Obj\_Ev < Max\_Obj\_Ev$ **then**
32: Break ;
33: End if
34: Update the current best solution g_*
35: **End while**

As shown in Algorithm 2, the main algorithmic structure of the conventional FPA is preserved in the proposed FPA-GA; the supplementary part is shown in gray. Indeed, the intervention of these two operators successively is controlled by the following condition; $rand < Prob\_CrMu$, where $rand$ is a random number $\in [0,1]$ and $Prob\_CrMu$ represents the probability of applying these latter.

Consequently, if this condition is checked two additional populations Pop_{Cross} and Pop_{Muta} will be added respectively. Then, an elite selection takes place to chose the n_f best solutions from the new global generated population $Pop_{Glob} = Pop \cup Pop_{Cross} \cup Pop_{Muta}$.

In addition, we note that Algorithm 2 contains two independent control structures form lines 22–24 and lines 31–33, which their purpose is to avoid execution of extra objective function evaluations by the algorithm. It is also to be noted that the number of objective function evaluations $Numb\_Obj\_Ev$ is always incremented while $Numb\_Obj\_Ev < Max\_Obj\_Ev$ by n_f (see line 21 and 29).

It should be noted that, the strong point of our algorithm resides in the fact that it has a simple algorithmic structure compared with other algorithms, which makes its implementation very easy.

5 Experimental Study

In this section, the FPA-GA algorithm is benchmarked on 25 benchmark functions from a CEC2005 special session (Suganthan *et al*, 2005). The benchmark functions used are minimization functions. They can be divided into four groups: unimodal, multimodal, fixed-dimension multimodal, and composite functions (Suganthan *et al*, 2005). The FPA-GA algorithm was run 20 times on each benchmark function. The number of decision variables is N. For each algorithm (FPA-GA and all other algorithms used in comparative study; FPA, MGOFPA, CMA-ES, CLPSO, JADE, jDE and CoDE and each test function, 20 independent runs were conducted with $n \times 100000$ function evaluations. In our experimental studies, the average and standard deviation ($Mean$ and $Std\ Div$) of the function error value $\left(f(\vec{x}) - f(\vec{x}^*)\right)$ were recorded for measuring the performance of each algorithm, where $\vec{x}$ is the best solution found by the algorithm in a run and $\vec{x}^*$ is the global optimum of the test function. All obtained results are given in Table 1 where the best results are marked in gray spaces. Moreover, $Wilcoxon's$ rank-sum statistical test was carried out at 5% significance level to judge whether the results of FPA-GA algorithm differ from those of the other algorithms in a statistically significant way. In addition, $\ominus$ indicates that FPA-GA performs significantly better than the tested algorithm on the specified function a $\oplus$ indicates that FPA-GA performs not as good as the tested algorithm, and a $\odot$ means that the Wilcoxon rank sum test cannot distinguish between the simulation results of FPA-GA and the tested algorithm. All Wilcoxon rank-sum based comparison of different obtained results are summarized in Table 2.

In all simulation tests, we have adapted FPA-GA, FPA and MGOFPA respectively with the following parameters combination: $p = 0.2, n_f = 50$, $Prob\_CrMu = 0.1$ $Prob\_GOB = 0.1$ (Draa, 2015), $p_C = 0.55$, the mutation rate p_m is given by: $p_m = 1 - \frac{0.1}{N_b} \times i, i = 1, \dots \dots, N_b$ is used in the following condition $p_m < p_{m\,rand}$, where $p_{m\,rand}$ is a random number. Furthermore, in this paper we have used the

following crossover and mutation kinds: arithmetical crossover (Michalewicz, 1992) and non-uniform mutation (Michalewicz, 1992).

Consequently, we can observe clearly from these two tables that FPA-GA performed better than all other algorithm. More precisely (see Table 2), the FPA-GA performed better than FPA, MGOFPA, CMA-ES, CLPSO, JADE, jDE and CoDE in 24, 24, 21, 23, 20, 23 and 21 cases (functions) respectively out of 25 and equal to these latter in 1 ,1,1,1,2,1,2 cases out of 25. Also, FPA-GA performs worse in 3, 1,3,1,2 cases out of 25 than CMA-ES, CLPSO, JADE, jDE and CoDE respectively.

It is clear from this simple presentation, that adding these two operators (crossover and mutation) to the main algorithmic FPA structure allows to improve significantly its performance. Thus, this is due primarily to an improvement of the diversity of the population (three sub-populations Pop, Pop_{Cross} and Pop_{Muta}) which greatly increases the chance to find the best solution, and also to an enhancement of the intensification ability by the association of these two operators and the elite selection mechanism.

6 Conclusions

A new hybrid optimisation method named FPA-GA is introduced in this paper, which considerably improves the performance of the original FPA algorithm by integrating the conventional FPA with two GAs main operators; crossover and mutation. In FPA-GA, the aim of using these latter is to improve the diversification and the intensification characteristics. The experimental studies were carried out on 25 global numerical optimization problems used in the CEC2005 special session on real-parameter optimization. FPA-GA was compared with; the standard FPA, a new FPA variant called MGOFPA, the CMA-ES, the CLPSO, and three DE variants called respectively JADE, jDE and CoDE. The obtained experimental results shown clearly that FPA-GA performances are better than the seven competitors.

The proposed FPA-GA algorithm should be used to solve multi-objective optimization problems in the future to validate its performance. In addition, there exists many NP- hard problems in literature, such as traveling salesman problem, graph-coloring problem, finder of polynomials based on root moments (Huang *et al*, 2004) and knapsack problem. In order to test performance of FPA-GA comprehensively, it should be used to solve these NP-hard problems in the future.

References

C. Blum and X. Li. Swarm intelligence in optimization. In *C. Blum, D. Merkle (Eds.), Swarm Intelligence: Introduction and Applications*, Springer Verlag, Berlin, pages 43-86, 2008.

J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer. Selfadapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Trans. Evolut. Comput.,* 10(6):646–657,2006.

A. Draa. On the performances of the flower pollination algorithm- qualitative and quantitative analyses. *Appl. Soft Comput,* 34: 349–371,2015.

J. Derrac, G. Molina, and F. Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evolut Comput*, 1:3–18, 2011.

J. H. Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control and artificial intelligence*. *MIT Press*, 1992.

D.S. Huang, H.H.S. Ip, and Z. Chi. A neural root finder of polynomials based on root moments. *Neural Comput*. 16(8):1721–1762, 2004.

N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolut. Comput.,* 9(2):159–195, 2001.

Z.W. Geem, J.H. Kim, and G.V. Loganathan. A new heuristic optimization algorithm harmony search. *Simulation*, 76(2):60–68, 2001.

J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, Perth, Australia, pages 1942–1948, 1995.

J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Trans. Evolut. Comput.,* 10(3): 281–295, 2006.

Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*, *Springer-Verlag*, New York, 1992.

A. Mucherino and O. Seref. Monkey search: a novel metaheuristic search for global optimization. In *Proceedings of the American Institute of Physics Conference*, USA, pages 162–173,2007.

I. Pavlyukevich. L´evy flights, non-local search and simulated annealing. *J. Computational Physics*, 226: 1830–1844, 2007.

K. Sastry, D. Goldberg, and G. Kendall. Genetic algorithms. In *E.K. Burke and G. Kendall (eds.). Introductory Tutorials in Optimisation, Decision Support and Search Methodology*. ISBN: 0387234608, Springer. Chapter 4, pages 97-125, 2005.

K. Socha and M. Dorigo. Ant colony optimization for continuous domains. *Eur. J. Op. Res,* 185(3): 1155–1173, 2008.

R. Storn and K. Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim,*11:341–359, 1997.

P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari. *Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization*. Nanyang Technol. Univ., Singapore, Tech. Rep. KanGAL #2005005, IIT Kanpur, India, May 2005.

Y. Xinshe. Multiobjective firefly algorithm for continuous optimization. *Eng. Comput*. 29(2): 175–184, 2012.

L. Z. Yongquan and Z. G. Jiakun. Leader glowworm swarm

optimization algorithm for solving nonlinear equations systems. *Electr. Rev*. 88(1b): 101–106, 2012.

X.S. Yang and S. Deb. Cuckoo search via Lévy flights. In *Proceedings of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009, India),* IEEE Publications, USA, pages 210–214,2009.

X.S. Yang. A new metaheuristic bat-inspired algorithm. In *J.R. Gonzalez, D. A. Pelta, C. Cruz (Eds.), Nature Inspired Cooperative Strategies for Optimization.* Springer-Ver-lag, Berlin, Germany, pages 65–74, 2010.

X. S. Yang, Z. Cui, R. Xiao, A. H. Gandomi, and M. Karamanoglu. *Swarm Intelligence and Bio-Inspired Computation*. Elsevier, *Waltham, MA*, 2013.

X.S. Yang. Flower pollination algorithm for global optimization. In *Unconventional Computation and Natural Computation,* Springer, Berlin, pages 240–249, 2012.

Y. Wang, Z. Cai, and Q. Zhang. Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Trans. Evol. Comput.*, 15(1): 55–66, Feb. 2011.

J. Zhang and A. C. Sanderson. JADE: adaptive differential evolution with optional external archive. *IEEE Trans. Evolut. Comput.,* 13(5):945-958, 2009.

Table 1. Experimental Results of FPA, MGOFPA, CMA-ES, CLPSO, JADE, jDE, CoDE and FPA-GA over 20 Independent runs on 25 test functions of $n = 30$ variables with 100000 Max_Obj_Ev.

Function	FPA-GA		FPA		MGOFPA		CMA-ES		CLPSO		JADE		jDE		CoDE	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
1	0.00e+00	0.00e+00	4.03e-29	1.08e-28	1.26e-29	4.59e-29	1.80e-25	4.65e-26	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
2	3.38e-11	7.54e-11	3.44e-18	8.02e-18	1.55e-02	1.79e-02	6.37e-25	1.78e-25	8.12e+02	2.32e+02	9.37e-29	1.07e-28	8.84e-07	1.12e-06	1.77e-15	2.19e-15
3	3.77e-10	1.65e-10	1.38e+06	1.87e+06	1.27e+06	8.95e+05	5.13e-21	1.30e-21	1.70e+07	2.61e+06	6.57e+03	3.64e+03	2.02e+05	9.92e+04	1.08e+05	5.38e+04
4	4.66e-06	1.15e-06	2.46e-04	4.79e-04	3.73e-01	3.87e-01	6.11e+05	1.68e+06	6.79e+03	1.10e+03	2.56e-14	8.55e-14	3.07e-02	5.68e-02	8.09e-03	2.24e-02
5	1.64e-04	1.27e-04	5.92e+01	2.21e+01	4.80e+01	7.00e+00	3.35e-10	8.62e-11	4.13e+03	4.76e+02	7.89e-06	3.52e-05	5.65e+02	5.22e+02	5.14e+02	4.42e+02
6	1.07e-11	4.12e-12	2.16e+01	4.26e+01	1.75e+01	2.00e+01	3.98e-01	1.22e+00	5.90e+00	1.27e+01	1.00e+01	2.85e+01	2.47e+01	2.69e+01	7.39e-10	1.99e-09
7	8.32e-05	5.38e-06	1.85e-02	1.43e-02	2.68e-02	3.52e-02	1.81e-03	4.39e-03	4.48e-01	8.44e-02	8.17e-03	7.32e-03	1.19e-02	7.76e-03	7.41e-03	8.51e-03
8	4.67e-02	1.46e-04	2.10e+01	9.79e-02	2.10e+01	7.65e-02	2.04e+01	6.89e-01	2.09e+01	5.05e-02	2.09e+01	6.48e-02	2.09e+01	3.31e-02	2.01e+01	1.05e-01
9	1.76e-03	1.42e-04	5.27e+01	2.38e+01	2.55e+01	9.76e+00	4.00e+02	1.15e+02	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
10	1.13e-03	1.47e-04	1.15e+02	8.63e+01	4.36e+01	3.58e+01	4.41e+01	1.49e+01	1.04e+02	1.77e+01	2.25e+01	2.96e+00	5.33e+01	8.70e+00	3.82e+01	1.14e+01
11	2.00e-02	1.04e-03	3.29e+01	8.41e+00	2.12e+01	7.49e+00	6.72e+00	2.23e+00	2.60e+01	1.72e+00	2.54e+01	2.27e+00	2.76e+01	1.46e+00	1.32e+01	3.84e+00
12	2.55e-02	1.54e-03	4.19e+01	7.61e+00	3.19e+01	8.43e+00	1.36e+04	1.42e+04	1.95e+04	5.56e+03	6.30e+03	7.21e+03	6.05e+03	5.30e+03	2.63e+03	1.91e+03
13	5.83e-04	2.58e-04	7.50e+00	7.29e+00	3.91e+00	2.97e+00	3.22e+00	8.63e-01	2.10e+00	2.21e-01	1.51e+00	6.68e-02	1.68e+00	1.21e-01	1.56e+00	3.17e-01
14	6.88e-02	7.40e-04	1.34e+01	5.76e-01	1.35e+01	2.98e-01	1.47e+01	2.33e-01	1.27e+01	2.28e-01	1.22e+01	2.76e-01	1.29e+01	2.20e-01	1.24e+01	4.91e-01
15	7.51e-04	4.35e-05	2.53e+02	8.60e+01	3.00e+02	1.12e+02	3.67e+02	2.10e+02	6.12e+01	4.10e+01	3.51e+02	1.14e+02	3.89e+02	8.78e+01	4.00e+02	7.94e+01
16	7.87e-04	1.05e-04	2.66e+02	1.15e+02	9.27e+01	8.02e+01	3.65e+02	3.10e+02	1.70e+02	3.47e+01	1.28e+02	1.42e+02	7.81e+01	2.23e+01	6.45e+01	1.64e+01
17	7.58e-04	1.29e-04	1.90e+02	1.29e+02	2.68e+02	8.71e+01	4.97e+02	3.47e+02	2.54e+02	4.25e+01	1.18e+02	1.00e+02	1.46e+02	4.27e+01	6.51e+01	1.27e+01
18	6.82e-04	2.44e-05	8.26e+02	2.01e+00	9.04e+02	1.07e+00	9.03e+02	2.12e-01	9.14e+02	1.34e+00	9.03e+02	2.61e-01	9.04e+02	1.23e+00	9.04e+02	9.64e-01
19	6.86e-04	2.96e-05	8.25e+02	1.77e+00	2.12e+02	3.13e+00	9.03e+02	2.12e-01	9.09e+02	2.20e+01	9.04e+02	1.06e+00	9.04e+02	1.04e+00	9.04e+02	1.04e+00
20	6.96e-04	3.14e-05	8.25e+02	2.28e+00	9.03e+02	7.08e-01	9.03e+02	2.99e-01	9.12e+02	8.11e+00	9.04e+02	7.62e-01	9.04e+02	1.07e+00	9.04e+02	1.34e+00
21	6.23e-04	1.69e-05	7.39e+02	1.80e+02	5.33e+02	1.32e+02	5.00e+02	2.63e-12	5.00e+02	1.25e-12	5.00e+02	5.05e-14	5.00e+02	3.91e-14	5.00e+02	8.65e-14
22	4.71e-04	4.94e-05	5.08e+02	4.57e+00	8.74e+02	2.22e+01	8.19e+02	1.30e+01	9.64e+02	1.05e+01	8.66e+02	2.05e+01	8.74e+02	1.54e+01	8.58e+02	2.23e+01
23	6.31e-04	2.32e-05	7.88e+02	1.83e+02	5.90e+02	1.73e+02	5.36e+02	3.89e+00	5.34e+02	2.04e-04	5.54e+02	9.00e+01	5.34e+02	2.59e-04	5.34e+02	4.51e-04
24	6.17e-04	1.53e-05	2.12e+02	3.13e+00	5.77e+02	3.57e+02	2.00e+02	6.18e-14	2.00e+02	1.46e-12	2.00e+02	2.91e-14	2.00e+02	2.91e-14	2.00e+02	2.91e-14
25	5.05e-04	1.05e-05	2.16e+02	6.91e-01	1.56e+03	1.09e+01	2.10e+02	6.05e+00	2.00E+02	1.96E+00	2.13e+02	7.95e-01	2.11e+02	7.31e-01	2.13e+02	9.12e-01

Table 2. Summarised Wilcoxon rank-sum comparisons between the proposed algorithm as reference and FPA, MGOFPA, CMA-ES, CLPSO, JADE, jDE, CoDE

Functions	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	⊖	⊙	⊕
Algorithms																												
FPA	⊖	⊙	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	**24**	**1**	**0**
MGOFPA	⊙	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	**24**	**1**	**0**
CMA-ES	⊙	⊕	⊕	⊖	⊕	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	**21**	**1**	**3**
CLPSO	⊙	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊕	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	**23**	**1**	**1**
JADE	⊙	⊙	⊖	⊕	⊕	⊖	⊖	⊖	⊕	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	**20**	**2**	**3**
jDE	⊙	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊕	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	**23**	**1**	**1**
CoDE	⊙	⊕	⊖	⊖	⊖	⊙	⊖	⊖	⊕	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	**21**	**2**	**2**

DOI: 10.3384/ecp171421060