Neural Comput & Applic (2015) 26:1257–1263
DOI 10.1007/s00521-014-1806-7

ORIGINAL ARTICLE

Evolutionary population dynamics and grey wolf optimizer

Shahrzad Saremi · Seyedeh Zahra Mirjalili · Seyed Mohammad Mirjalili

Received: 7 July 2014 / Accepted: 8 December 2014 / Published online: 31 December 2014

Abstract Evolutionary population dynamics (EPD) deal with the removal of poor individuals in nature. It has been proven that this operator is able to improve the median fitness of the whole population, a very effective and cheap method for improving the performance of meta-heuristics. This paper proposes the use of EPD in the grey wolf optimizer (GWO). In fact, EPD removes the poor search agents of GWO and repositions them around alpha, beta, or delta wolves to enhance exploitation. The GWO is also required to randomly reinitialize its worst search agents around the search space by EPD to promote exploration. The proposed GWO–EPD algorithm is benchmarked on six unimodal and seven multi-modal test functions. The results are compared to the original GWO algorithm for verification. It is demonstrated that the proposed operator is able to significantly improve the performance of the GWO algorithm in terms of exploration, local optima avoidance, exploitation, local search, and convergence rate.

Keywords Grey wolf optimizer · Optimization · Evolutionary algorithms · Heuristic

S. Saremi (✉)
School of Information and Communication Technology, Nathan Campus, Griffith University, Brisbane, QLD 4111, Australia
e-mail: shahrzad.saremi@griffithuni.edu.au

S. Saremi
Queensland Institute of Business and Technology, Mt Gravatt, Brisbane, QLD 4122, Australia

S. Z. Mirjalili · S. M. Mirjalili
Zharfa Pajohesh System (ZPS) Co., Unit 5, NO. 30, West 208 St., Third Sq. Tehranpars, P.O. Box 1653745696, Tehran, Iran

1 Introduction

Over the last two decades, nature-inspired optimization techniques have become a reliable alternatives compared to other optimization approached. As their name implies, such methods mostly mimic natural phenomena for optimization. The majority of the nature-inspired algorithms are classified as stochastic optimization techniques. The term stochastic stands opposite of deterministic.

In deterministic optimization, the improvement of variables of a given problem results in a similar solution every time. However, stochastic algorithms find different solutions for optimization problems in every run. The general framework of stochastic optimization algorithm is that they start optimization process with one of a set of random solutions. The candidate solution(s) is then improved based on the mechanism of the algorithm. This proves is iteratively run until the satisfaction of an end criterion.

Due to the advantages of stochastic optimization paradigms, such high local optima avoidance, high convergence speed, simplicity, and derivative-free mechanism, several algorithms have been proposed recently. Some of the most popular are genetic algorithm (GA), particle swarm optimization (PSO), ant colony optimization (ACO), and differential evolution (DE). There are also a massive number of recently proposed algorithms in the literature: gravitational local search (GLSA) [1], big bang–big crunch (BBBC) [2], gravitational search algorithm (GSA) [3], central force optimization (CFO) [4], artificial chemical reaction optimization algorithm (ACROA) [5], black hole (BH) [6] algorithm, small-world optimization algorithm (SWOA) [7], galaxy-based search algorithm (GbSA) [8], curved space optimization (CSO) [9], biogeography-based optimizer (BBO) [10], marriage in honey bees optimization algorithm (MBO) in 2001 [11], artificial

fish-swarm algorithm (AFSA) in 2003 [12], termite algorithm in 2005 [13], wasp swarm algorithm in 2007 [14], monkey search in 2007 [15], bee collecting pollen algorithm (BCPA) in 2008 [16], Cuckoo search (CS) in 2009 [17], dolphin partner optimization (DPO) in 2009 [18], firefly algorithm (FA) in 2010 [19], bird mating optimizer (BMO) in 2012 [20], and fruit fly optimization algorithm (FOA) in 2012 [21].

The large number of application of stochastic optimization techniques in science and engineering are also another evidence of popularity and applicability of these algorithms. One of the ways of improving the performance of algorithms in this field is hybridization and combining different operators. In this case, different operators and mechanisms of two or more algorithms are combined. Some of the examples are

- Section operator of GA was integrated to PSO in [22].
- Crossover and mutation of GA was applied to PSO in [23].
- Social thinking of PSO was integrated to GSA in [24].
- The mutation operators of DE were embedded to PSO in [25].
- The pheromone updating rules of ACO were incorporated to PSO in [26].
- ABC was equipped with the personal best concepts of PSO in [27].
- ABC was equipped with the global best concept of PSO in [28].
- Chaotic mutation operator for PSO in [29].
- Selection, crossover, and mutation of GA was integrated to ACO in [30].
- Mutation of DE was embedded to ACO pheromone trail in [31].

Selection, combination, and mutation are the most widely used evolutionary operators that have been applied to many other meta-heuristic. However, such operators only manipulated individuals. According to Lewis et al. [32], there is one more evolutionary operator called evolutionary population dynamics (EPD) that considers and manipulates the population as a whole. The main inspiration of this operator is the theory of self-organizing critically (SOC) [33]. According to Bak [34], critical state in nature is "the most efficient state that can actually be reached dynamically". In a balanced population, for instance, small mutations (perturbations) provide delicate balances in the population without external force [32]. Extremal optimization (EO) [35] is a meta-heuristic inspired by the Bak–Sneppen model of self-organized criticality and based on EPD. In this algorithm, the worst individuals in the population are omitted compared to GA where best individuals are combined.

Evolutionary programming using self-organizing criticality (EPSCO) is another SCO-based and was proposed by Lewis et al. [36]. According to Randall and Lewis [37], this algorithm can be considered as the improved method of Fogel [38] with an extra selection operator from Bak–Sneppen model.

The literature [39–45] shows that EO is a very powerful optimization technique. Undoubtedly, this is due to the EPD operator of this algorithm that eliminates the worst individuals and improves them. Despite the significant merits of EPD operator, there is currently little in the literature about the use of this operator in different stochastic optimization techniques. This motivates our attempts to apply the EPD to our recently proposed grey wolf optimizer (GWO) and investigate its effectiveness. The rest of this paper is structured in the following way.

Section 2 discusses the concepts of EPD and EO algorithm. Section 3 discusses the preliminary concepts of the GWO algorithm. The method of applying EPD–GWO is proposed in Sect. 3.2. Section 4 presents results, discussion, and analysis. Eventually, Sect. 5 provides the conclusion of this work and recommends some directions for future studies.

2 Evolutionary population dynamics and extremal optimization

The main foundation of EPD is based on a theory of SOC [33]. According to Bak [34], critical state in nature is "the most efficient state that can actually be reached dynamically". According to the SOC theory, small mutations (perturbations) provide delicate balances in a balanced population without external force [32]. In the evolution of different species, it is observed that evolution also applies on the poor species. In this case, the total population of a species are effected by the removal of the poor individuals. The process of eliminating poor individuals in a population is called EPD as shown in Fig. 1.

Extremal optimization (EO) [35] is a meta-heuristic inspired by the Bak–Sneppen model of self-organized

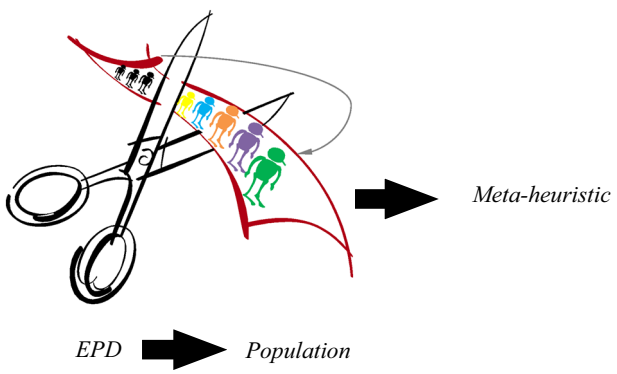

Fig. 1 EPD eliminates poor individuals in a population

criticality and EPD. In this algorithm, the worst individuals in the population are omitted compared to GA where best individuals are combined. Evolutionary programming using self-organizing criticality (EPSCO) is another SCO-based and was proposed by Lewis et al. [36]. The main reason of the success of EPD is that it improved the median of the population by removing the worst individuals. The removal of worst individuals is the first step when using EPD in a population-based algorithm. The next step is to mutate or re-position the removed individuals around the best solutions.

In the next section, we propose a method to integrate the EPD operator to the recently proposed GWO algorithm.

3 Evolutionary population dynamics for grey wolf optimizer

This section first briefly presents the GWO algorithm and then describes the proposed method.

3.1 Grey wolf optimizer (GWO)

The GWO algorithm was proposed by Mirjalili et al. [46]. It mimics the hunting behavior and social leadership of grey wolves in nature. Similarly to other meta-heuristics, it initials the optimization process by generating a set of random candidate solutions. In every iteration, the three best candidate solutions are considered as alpha, beta, and delta wolves, who take the lead toward to promising regions of the search space. The rest of grey wolves are assumed as omega and required to encircle alpha, beta, and delta with the hope to find better solutions. The mathematical formulation of omega wolves are as follows [46]:

$$\vec{D} = \left| \vec{C} \cdot \overrightarrow{X_{\mathrm{p}}}(t) - \vec{X}(t) \right| \tag{3.1}$$

$$\vec{X}(t+1) = \overrightarrow{X_{\mathrm{p}}}(t) - \vec{A} \cdot \vec{D} \tag{3.2}$$

where t indicates the current iteration, $\vec{A} = 2\vec{a} \cdot \overrightarrow{r_1} - \vec{a}, \vec{C} = 2 \cdot \overrightarrow{r_2}, \overrightarrow{X_{\mathrm{p}}}$ is the position vector of the prey, $\vec{X}$ indicates the position vector of a grey wolf, $\vec{a}$ is linearly decreased from 2 to 0, and r_1, r_2 are random vectors in [0, 1].

It should be noted that each mega wolf is required to update its position with respect to alpha, beta, and delta simultaneously as follows [46]:

$$\overrightarrow{D_\alpha} = \left| \vec{C}_1 \cdot \overrightarrow{X_\alpha} - \vec{X} \right| \tag{3.3}$$

$$\overrightarrow{D_\beta} = \left| \vec{C}_2 \cdot \overrightarrow{X_\beta} - \vec{X} \right| \tag{3.4}$$

$$\overrightarrow{D_\delta} = \left| \overrightarrow{C_3} \cdot \overrightarrow{X_\delta} - \vec{X} \right| \tag{3.5}$$

$$\overrightarrow{X_1} = \overrightarrow{X_\alpha} - \vec{A}_1 \cdot \left(\overrightarrow{D_\alpha} \right) \tag{3.6}$$

$$\overrightarrow{X_2} = \overrightarrow{X_\beta} - \overrightarrow{A_2} \cdot \left(\overrightarrow{D_\beta} \right) \tag{3.7}$$

$$\overrightarrow{X_3} = \overrightarrow{X_\delta} - \overrightarrow{A_3} \cdot \left(\overrightarrow{D_\delta} \right) \tag{3.8}$$

$$\vec{X}(t+1) = \frac{\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3}}{3} \tag{3.9}$$

It was argued by Mirjalili et al. that the parameters A and C oblige the GWO algorithm to explore and exploit the search space. Half of the iterations is devoted to exploration (when $|A| > 1$) and the rest to exploitation (when $|A| < 1$). The parameter C also changed randomly to resolve local optima stagnation during the course of optimization.

3.2 EPD for GWO

The mechanism of EPD is very straightforward for population-based algorithm as discussed by Lewis et al. [32]. Since the GWO algorithm in a stochastic population-based algorithm, fortunately, the EPD is readily applicable to this algorithm. As discussed in the introduction, EPD offers elimination of worst individuals in a population instead on evolving the best ones. In order to equip GWO with EPD, we eliminate half of the worst search agent in each iteration and re-initialize them in four random positions with equal probability as follows:

1. Reposition the poor candidate solution around the position of alpha

   $$\vec{X}(t+1) = \overrightarrow{X_\alpha}(t) \pm (ub - lb \cdot r + lb) \tag{3.10}$$

   where ub is the upper bound of the search space, lb indicates the lower bound of the search space, r is a random number in [0, 1].
2. Reposition the poor candidate solution around the position of beta

   $$\vec{X}(t+1) = \overrightarrow{X_b}(t) \pm (ub - lb \cdot r + lb) \tag{3.11}$$

   where ub is the upper bound of the search space, lb indicates the lower bound of the search space, r is a random number in [0, 1].
3. Reposition the poor candidate solution around the position of delta

   $$\vec{X}(t+1) = \overrightarrow{X_d}(t) \pm (ub - lb \cdot r + lb) \tag{3.12}$$

   where ub is the upper bound of the search space, lb indicates the lower bound of the search space, r is a random number in [0, 1].
4. Reposition the poor candidate solution in a random place within the boundaries of the search space

```
Initialize the grey wolf population X_i (i = 1, 2, ..., n)
Initialize a, A, and C
Calculate the fitness of each search agent
X_α=the best search agent
X_β=the second best search agent
X_δ=the third best search agent
while (t < Max number of iterations)
    for each search agent
            Update the position of the current search agent by equation (3.7)
    end for
    Update a, A, and C
    Calculate the fitness of all search agents
    Update X_α, X_β, and X_δ
    t=t+1
    Sort the population based on the fitness
    for i=(n/2)+1 to n
            Update the position of i-th wolf  based on (3.10), (3.11), (3.12), or (3.13)
    end for
end while
return X_α
```

Fig. 2 Pseudo codes of the GWO-EPD algorithm

$$\vec{X}(t+1) = (ub - lb \cdot r + lb) \tag{3.13}$$

where ub is the upper bound of the search space, lb indicates the lower bound of the search space, r is a random number in [0, 1].

The motivation of the first three proposed rules is to improve the median of whole population in each iteration. However, the last rule re-positions the candidates randomly around the search space, a very helpful mechanism to promote exploration and resolving local optima stagnation. The pseudo codes of the proposed GWO–EPD algorithm are shown in Fig. 2.

4 Results and discussion

This section experimentally investigates the effectiveness of the proposed EPD-based GWO algorithm. Similarly to other works in the literature, we employ several test functions to benchmark the performance of the proposed algorithm [47–50]. The selected test functions are presented in Tables 1 and 2 [51–55]. As may be seen in these two tables, the test functions are divided to two groups: unimodal and multi-modal [56–58]. We deliberately chose this set of test functions to benchmark exploitation and exploration, respectively. Unimodal test functions are suitable for testing the exploitation, whereas multi-modal test functions are appropriate for examining exploration of an algorithm.

The proposed algorithm is run 10 times, and we report average and standard deviation of the best obtained solutions in the last iteration in Tables 3 and 4. The results are also compared with GWO for verification. Note that we utilize 50 search agents and 500 iterations for solving each test function.

The results of the algorithms on the unimodal test functions show that the GWO–EPD algorithm is able to provide better results compared to the GWO algorithm. The results are significantly superior occasionally. Since the unimodal test functions have only one global optimum, these results show that the exploitation, local search, and convergence rate of the GWO is improved by EPD. This is due to the fact that the poor individuals are repositioned around the best solutions obtained so far at each iteration: alpha, beta, and delta wolves. This mechanism improves the local search, exploitation of promising solutions, and converge of search agents toward alpha, beta, and delta.

The results of the multi-modal test functions are presented in Table 4.

This table shows that the GWO–EPD algorithm outperforms the GWO algorithm on the majority of the multi-modal test functions. The results are again significantly different occasionally. The reason of the better performance of the GWO–EPD is the reposition of search agents randomly around search space based of Eq. (3.13). This mechanism highly promotes exploration and local optima avoidance. Since multi-modal functions have a massive number of local solution, the results show that the proposed EPD-based GWO algorithm handles such difficulties easier than the GWO algorithm.

To sum up, the results show that the EPD algorithm is able to significantly improve the performance of the GWO algorithm. On one side, repositioning search agent around alpha, beta, and delta boosts exploitation, local search, and

Table 1 Unimodal benchmark functions

Function	Dim	Range	$f_{\min}$
$f_1(x) = \sum_{i=1}^{n} x_i^2 + 30$	30	[−100, 100]	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert + 3$	30	[−10, 10]	0
$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j \right)^2 + 30$	30	[−100, 100]	0
$f_4(x) = \max_i \{ \lvert x_i \rvert, 1 \le i \le n \} + 30$	30	[−100, 100]	0
$f_5(x) = \sum_{i=1}^{n-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + (x_i - 1)^2 \right] + 15$	30	[−30, 30]	0
$f_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2 + 75$	30	[−100, 100]	0
$f_7(x) = \sum_{i=1}^{n} i x_i^4 + \text{random}[0, 1) + 0.25$	30	[−1.28, 1.28]	0

Table 2 Multi-modal benchmark functions

Function	Dim	Range	$f_{\min}$
$F_8(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right) + 300$	30	$[-500, 500]$	-418.9829×5
$F_9(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right] + 2$	30	$[-5.12, 5.12]$	0
$F_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e + 10$	30	$[-32, 32]$	0
$F_{11}(x) = \frac{1}{4{,}000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 + 400$	30	$[-600, 600]$	0
$F_{12}(x) = \frac{\pi}{n}\left\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2\left[1 + 10\sin^2(\pi y_{i+1})\right] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4) + 30$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	30	$[-50, 50]$	0
$F_{13}(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2\left[1 + \sin^2(3\pi x_i + 1)\right] + (x_n - 1)^2\left[1 + \sin^2(2\pi x_n)\right]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4) + 30$	30	$[-50, 50]$	0

Table 3 Results on unimodal functions

Test function	GWO–EPD	GWO
F1	1.6915 ± 0.6126	86.2764 ± 463.9271
F2	54.8619 ± 23.4144	55.5763 ± 24.6718
F3	5,795.515 ± 2599.173	5,603.5083 ± 2463.6797
F4	1.1309 ± 0.2842	1.0276 ± 0.2583
F5	798.0715 ± 794.6384	651.1185 ± 767.3522
F6	149.0123 ± 806.9220	160.3304 ± 869.2208

Table 4 Results on multi-modal functions

Test function	GWO–EPD	GWO
F7	0.2440 ± 0.2653	0.2462 ± 0.2277
F8	−12,702.47 ± 932.4343	−12,465.01 ± 1,362.617
F9	147.8416 ± 54.1366	169.9558 ± 61.9270
F10	20.8627 ± 0.0876	20.8604 ± 0.0740
F11	1.9205 ± 5.3332	1.9207 ± 5.3631
F12	0.1422 ± 0.1634	0.1623 ± 0.1695
F13	0.0825 ± 0.0314	0.0967 ± 0.0346

convergence. On the other side, re-initializing search agents in random position throughout the search space promotes exploration and local optima avoidance of the GWO algorithm.

5 Conclusion

This paper proposed the use of the EPD operator in the recently proposed GWO algorithm. Four rules were introduced to integrate EPD–GWO. In the proposed method, half of the search agents were re-positioned around alpha, beta, delta, and random positions in the search space during optimization. Thirteen standard test functions were employed to benchmark the performance of the GWO–EPD algorithm. The statistical results were compared to the GWO algorithm. It was found that the EPD operator is able to improve the performance of the GWO algorithm on both unimodal and multi-modal test functions. According to the findings, the following conclusion can be made:

- EPD improved the median fitness of the whole population over the course of iterations.
- EPD has the potential to improve exploration and local optima avoidance of the search space if the worst individuals are repositioned randomly around the search space.
- EPD has the potential to improve exploitation, local search, and convergence toward the promising regions of the search space if the worst individuals are repositioned around the best solutions obtained so far.

For future work, it is recommended to investigate the effectiveness of the EPD operator in enhancing the performance of other meta-heuristics.

References

1. Webster B, Bernhard PJ (2003) A local search optimization algorithm based on natural principles of gravitation. In:

Proceedings of the 2003 international conference on information and knowledge engineering (IKE'03), Las Vegas, NV, USA, 2003, pp 255–261
2. Erol OK, Eksin I (2006) A new optimization method: big bang–big crunch. Adv Eng Softw 37:106–111
3. Rashedi E, Nezamabadi-Pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 179:2232–2248
4. Formato RA (2007) Central force optimization: a new metaheuristic with applications in applied electromagnetics. Prog Electromagn Res 77:425–491
5. Alatas B (2011) ACROA: artificial chemical reaction optimization algorithm for global optimization. Expert Syst Appl 38:13170–13180
6. Hatamlou A (2013) Black hole: a new heuristic optimization approach for data clustering. Inf Sci 222:175–184
7. Du H, Wu X, Zhuang J (2006) Small-world optimization algorithm for function optimization. In: Advances in natural computation. Springer, pp 264–273
8. Shah-Hosseini H (2011) Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimisation. Int J Comput Sci Eng 6:132–140
9. Moghaddam FF, Moghaddam RF, Cheriet M (2012) Curved space optimization: a random search based on general relativity theory. arXiv preprint arXiv:1208.2214
10. Simon D (2008) Biogeography-based optimization. IEEE Trans Evolut Comput 12:702–713
11. Abbass HA (2001) MBO: marriage in honey bees optimization—a haplometrosis polygynous swarming approach. In: Proceedings of the 2001 congress on evolutionary computation, 2001, pp 207–214
12. Li X (2003) A new intelligent optimization-artificial fish swarm algorithm. Doctor thesis, Zhejiang University of Zhejiang, China
13. Roth M (2005) Termite: a swarm intelligent routing algorithm for mobile wireless ad-hoc networks. Ph. D thesis, Cornel University
14. Pinto PC, Runkler TA, Sousa JM (2007) Wasp swarm algorithm for dynamic MAX-SAT problems. In: Adaptive and natural computing algorithms. Springer, pp 350–357
15. Mucherino A, Seref O (2007) Monkey search: a novel metaheuristic search for global optimization. In: AIP conference proceedings, p 162
16. Lu X, Zhou Y (2008) A novel global convergence algorithm: bee collecting pollen algorithm. In: Advanced intelligent computing theories and applications. With aspects of artificial intelligence. Springer, pp 518–525
17. Yang X-S, Deb S (2009) Cuckoo search via Lévy flights. In: World congress on nature and biologically inspired computing, 2009. NaBIC 2009, pp 210–214
18. Shiqin Y, Jianjun J, Guangxing Y (2009) A dolphin partner optimization. In: WRI global congress on intelligent systems, 2009. GCIS'09, pp 124–128
19. Yang X-S (2010) Firefly algorithm, stochastic test functions and design optimisation. Int J Bioinspired Comput 2:78–84
20. Askarzadeh A, Rezazadeh A (2013) A new heuristic optimization algorithm for modeling of proton exchange membrane fuel cell: bird mating optimizer. Int J Energy Res 37(10):1196–1204
21. Pan W-T (2012) A new fruit fly optimization algorithm: taking the financial distress model as an example. Knowl Based Syst 26:69–74
22. Abdel-Kader RF (2011) Hybrid discrete PSO with GA operators for efficient QoS-multicast routing. Ain Shams Eng J 2:21–31
23. Kao Y-T, Zahara E (2008) A hybrid genetic algorithm and particle swarm optimization for multimodal functions. Appl Soft Comput 8:849–857
24. Mirjalili S, Hashim SZM (2010) A new hybrid PSOGSA algorithm for function optimization. In: 2010 international conference on computer and information application (ICCIA), 2010, pp 374–377
25. Khamsawang S, Wannakarn P, Jiriwibhakorn S (2010) Hybrid PSO-DE for solving the economic dispatch problem with generator constraints. In: 2010 the 2nd international conference on computer and automation engineering (ICCAE), 2010, pp 135–139
26. Shuang B, Chen J, Li Z (2011) Study on hybrid PS-ACO algorithm. Appl Intell 34:64–73
27. El-Abd M (2011) A hybrid ABC-SPSO algorithm for continuous function optimization. In: 2011 IEEE symposium on swarm intelligence (SIS), 2011, pp 1–6
28. Zhu G, Kwong S (2010) Gbest-guided artificial bee colony algorithm for numerical function optimization. Appl Math Comput 217:3166–3173
29. Coelho LdS (2008) A quantum particle swarm optimizer with chaotic mutation operator. Chaos Solitons Fractals 37:1409–1418
30. Lee Z-J, Su S-F, Chuang C-C, Liu K-H (2008) Genetic algorithm with ant colony optimization (GA-ACO) for multiple sequence alignment. Appl Soft Comput 8:55–78
31. Duan H, Yu Y, Zhang X, Shao S (2010) Three-dimension path planning for UCAV using hybrid meta-heuristic ACO–DE algorithm. Simul Model Pract Theory 18:1104–1115
32. Lewis A, Mostaghim S, Randall (2008) Evolutionary population dynamics and multi-objective optimisation problems. In: Multi-objective optimization in computational intelligence: theory and practice, pp 185–206
33. Bak P, Tang C, Wiesenfeld K (1987) Self-organized criticality: an explanation of the 1/f noise. Phys Rev Lett 59:381
34. Bak P (1997) How nature works. Oxford University Press, Oxford
35. Boettcher S, Percus AG (1999) Extremal optimization: methods derived from co-evolution. arXiv preprint arXiv:math/9904056
36. Lewis A, Abramson D, Peachey T (2004) An evolutionary programming algorithm for automatic engineering design. In: Parallel processing and applied mathematics. Springer, pp 586–594
37. Randall M, Lewis A (2006) An extended extremal optimisation model for parallel architectures. In: Second IEEE international conference on e-science and grid computing, 2006. e-Science'06, pp 114–114
38. Fogel LJ (1962) Autonomous automata. Ind Res 4:14–19
39. Xie D, Luo Z, Yu F (2009) The computing of the optimal power consumption for semi-track air-cushion vehicle using hybrid generalized extremal optimization. Appl Math Model 33:2831–2844
40. Randall M (2007) Enhancements to extremal optimisation for generalised assignment. In: Progress in artificial life. Springer, pp 369–380
41. Randall M, Hendtlass T, Lewis A (2009) Extremal optimisation for assignment type problems. In: Biologically-inspired optimisation methods. Springer, pp 139–164
42. Gómez-Meneses P, Randall M, Lewis A (2010) A hybrid multi-objective extremal optimisation approach for multi-objective combinatorial optimisation problems. In: 2010 IEEE congress on evolutionary computation (CEC), 2010, pp 1–8
43. Tamura K, Kitakami H, Nakada A (2013) Distributed modified extremal optimization using island model for reducing crossovers in reconciliation graph. Eng Lett 21:81–88
44. Gomez Meneses PS (2012) Extremal optimisation applied to constrained combinatorial multi-objective optimisation problems. Ph. D thesis, Bond University
45. Tamura K, Kitakami H, Nakada A (2014) Island-model-based distributed modified extremal optimization for reducing crossovers in reconciliation graph. In: Transactions on engineering technologies. Springer, pp 141–156

46. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
47. Mirjalili S, Lewis A (2014) Adaptive gbest-guided gravitational search algorithm. Neural Comput Appl 25(7–8):1569–1584
48. Mirjalili S, Lewis A, Sadiq AS (2014) Autonomous particles groups for particle swarm optimization. Arab J Sci Eng 39(6):4683–4697
49. Mirjalili S, Mirjalili S, Yang X-S (2014) Binary bat algorithm. Neural Comput Appl 25:663–681
50. Mirjalili S, Wang G-G, Coelho LdS (2014) Binary optimization using hybrid particle swarm optimization and gravitational search algorithm. Neural Comput Appl 25(6):1423–1435
51. Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. IEEE Trans Evolut Comput 3:82–102
52. Digalakis J, Margaritis K (2001) On benchmarking functions for genetic algorithms. Int J Comput Math 77:481–506
53. Molga M, Smutnicki C (2005) Test functions for optimization needs. In: Test functions for optimization needs
54. Yang X-S (2010) Test problems in optimization. arXiv preprint arXiv:1008.0549
55. Mirjalili S, Lewis A (2013) S-shaped versus V-shaped transfer functions for binary particle swarm optimization. Swarm Evolut Comput 9:1–14
56. Saremi S, Mirjalili S, Lewis A (2014) How important is a transfer function in discrete heuristic algorithms. Neural Comput Appl 1–16. doi:10.1007/s00521-014-1743-5
57. Saremi S, Mirjalili S, Lewis A (2014) Biogeography-based optimisation with chaos. Neural Comput Appl 25(5):1077–1097
58. Saremi S, Mirjalili SM, Mirjalili S (2014) Chaotic krill herd optimization algorithm. Proc Technol 12:180–185