Eagle Strategy with Flower Algorithm

Xin-She Yang
School of Science and Technology
Middlesex University
Hendon Campus, London NW4 4BT
United Kingdom
(x.yang@mdx.ac.uk)

Suash Deb
Cambridge Institute of Technology
Cambridge Village, Tatisilwai
Ranchi-835103, Jharkhand
India
(suashdeb@gmail.com)

Xingshi He
School of Science
Xi'an Polytechnic University
No. 19 Jinhua South Road, Xi'an
P. R. China
(xsh1002@126.com)

***Abstract*—For any metaheuristic algorithm, it is important to balance exploration and exploitation because the interaction of these two key components can significantly affect the efficiency. However, how to achieve a fine balance is still an open problem. We attempt to explore this challenging issue by using eagle strategy in combination with the recently developed flower algorithm. Our results on test benchmarks suggest that more effort should focus on explorative search for multimodal problems.**

***Index Terms*—Eagle strategy, flower algorithm, optimization, nature-inspired algorithm, metaheuristic.**

I. Introduction

Nature-inspired metaheuristic algorithms, especially those based on swarm intelligence, form an important part of contemporary global optimization algorithms [3], [11], [10], [14], [20]. Good examples are particle swarm optimization [11], cuckoo search and firefly algorithm [14]. They work remarkably efficiently and have many advantages over traditional, deterministic methods and algorithms, and thus they have been applied in almost all areas of science, engineering and industry [8], [15], [20]. There are more than a dozen of swarm-based algorithms using the so-called swarm intelligence [12], [15], [16]. Efficient algorithms such as firefly algorithm and cuckoo search have been extended to solve multi-objective optimization problems [18], [22].

On the other hand, design optimization problems in engineering and industry are typically nonlinear. These design problems are under complex, often highly nonlinear constraints imposed by physics and design codes. As a result, such design problems are often NP-hard, and requires highly sophisticated tools. To find optimal solutions to such optimization problems is usually very challenging, if not impossible. A recent trend is to use nature-inspired metaheuristic algorithms to tackle such nonlinear global optimization problems. In fact, over the last two decades, about two dozen new algorithms have been developed, and they include particle swarm optimization, differential evolution, ant and bee algorithms, harmony search, firefly algorithm and cuckoo search. Intensive studies indicated that they have great potential in solving tough engineering optimization problems [8], [14].

The paper is organized as follows: First, we briefly highlight the key components of exploration and exploration in metaheuristics, followed by the analysis of possible balance based on the intermittent search theory. Then, we introduce briefly the eagle strategy and flower algorithm. Finally, we use these algorithms to solve two test benchmarks so as to confirm the optimal choice of parameters.

II. Exploration and Exploitation

Metaheuristics can be considered as a higher level of search strategy by trial and error to a complex problem, so as to find good solutions, and possibly optimal solutions, in a reasonably practical time. There is no guarantee that the best solutions can be found, and we even may not know whether an algorithm will work and why if it does work, though we may know the basic components that can help to work. The idea is to have an efficient but practical algorithm that will work most of the time and is able to produce good quality solutions. Among the quality solutions found so far, it is expected some of them are nearly optimal, though there is often no guarantee for such optimality.

Many metaheuristics have global convergence properties, and thus they usually can find the global optimality in practice within a relatively limited number of function iterations or evaluations. This makes it particular suitable for metaheuristic algorithms to solve global optimization. For example, cuckoo search uses not only a search technique with good convergence but also a randomization technique with more efficient Lévy flights [15].

For a metaheuristic algorithm to be efficient, it has to have some special capabilities. One of such is to be able to generate new solutions that can usually be more likely to improve the previous/existing solutions and also be able to cover most important search areas where the global optimum may lie. Another capability is that an algorithm should be able to escape any local optimum so that it cannot be stuck in a local mode. A good combination may lead to good efficiency under appropriate conditions, and this often requires to balance two important components of any metaheuristics: exploration and exploitation. However, this itself is an unresolved optimization task [3], [14], [15].

A. Fine Balance

Stochastic component in modern metaheuristics enables an algorithm to explore the search space and exploit search information, and such randomization techniques do not provide a way to balance exploration and exploitation automatically [3],

[14]. Consequently, parameter tuning by trial and error can be used to get the right values of algorithm-dependent algorithms; however, such tuning of parameters is a time-consuming task. The advantages of randomization enable an algorithm to have the ability to jump out of any local optimum so as to explore the search globally.

Though there is no theoretical framework, however, empirical knowledge from observations and simulations of the convergence behaviour of common optimization algorithms suggests that exploitation tends to increase the speed of convergence, while exploration tends to decrease the convergence rate of the algorithm. On the other hand, too much exploration increases the probability of finding the global optimality but with a reduced efficiency, while strong exploitation tends to make the algorithm being trapped in a local optimum. Therefore, there is a fine balance between the right amount of exploration and the right degree of exploitation. Despite its importance, there is no practical guideline for this balance. Consequently, each algorithm uses different degree of exploitation and exploration, often far from optimal [3], [14], [15]. Some algorithms may have intrinsically better balance among these two important components than other algorithms, and that is one of the reasons why some algorithms may perform better than others.

B. Optimality via Intermittent Search Theory

Even there is no guideline in practice, some preliminary work on the very limited cases exists in the literature and may provide some insight into the possible choice of parameters so as to balance these components [1], [2], [13]. Intermittent search strategies concern an iterative strategy consisting of a slow phase and a fast phase [1], [2]. Here the slow phase is the detection phase by slowing down and intensive, static local search techniques, while the fast phase is the search without detection and can be considered as an exploration technique. For example, the static target detection with a small region of radius a in a much larger spherical domain of radius R where $a \ll R$ can be modelled as a slow diffusive process in terms of random walks with a diffusion coefficient D.

Obviously, the optimal balance depends on the type of problems. If a problem is unimodal or convex, then there is no need to do much exploration. Aggressive exploitation is preferred. Intermittent switch between exploration stage and exploitation stage can be optimal for multimodal functions where the areas/volumes of the local modes are small, compared with the area/volume of the search domain. Thus, we are dealing with target modes with sparsity.

Let τ_a and τ_R be the mean times spent in intensive detection stage and the time spent in the exploration stage, respectively, in the 2D case [2]. The diffusive search process is governed by the mean first-passage time satisfying the following equations

$$D\nabla_r^2 t_1 + \frac{1}{2\pi\tau_a}\int_0^{2\pi}[t_2(r) - t_1(r)]d\theta + 1 = 0, \qquad (1)$$

$$\boldsymbol{u} \cdot \nabla_r t_2(r) - \frac{1}{\tau_R}[r_2(r) - t_1(r)] + 1 = 0, \qquad (2)$$

where t_2 and t_1 are times spent during the search process at slow and fast stages, respectively, and $\boldsymbol{u}$ is the average search speed [2].

After some lengthy mathematical analysis [1], [2], the optimal balance of these two stages can be estimated as

$$r_{\text{optimal}} = \frac{\tau_a}{\tau_R^2} \approx \frac{D}{a^2}\frac{1}{[2 - \frac{1}{\ln(R/a)}]^2}. \qquad (3)$$

An interesting observation is that the above results depend weakly on the domain size R, and thus balancing these two key components can lead to very efficient performance of the algorithm used [2]. In addition, increase the global exploration velocity u can also reduce the overall search time, and thus implicitly enhance the search efficiency of the algorithm.

It should be emphasized that the above result is only valid for 2D cases, and there is no general results for higher dimensions, except in some special 3D cases [1]. Now let us use this limited results to help choose the possible values of algorithm-dependent parameters in eagle strategy [19], as an example.

III. Eagle Strategy

Eagle strategy is a metaheuristic strategy for optimization, developed in 2010 by Xin-She Yang and Suash Deb [19]. More extensive studies have followed [21], [9]. It uses a combination of crude global search and intensive local search employing different algorithms to suit different purposes. In essence, the strategy first explores the search space globally using a Lévy flight random walk, if it finds a promising solution, then an intensive local search is employed using a more efficient local optimizer such as hill-climbing, differential evolution and/or other new algorithms. Then, the two-stage process starts again with new global exploration followed by a local search in a new region.

The advantage of such a combination is to use a balanced tradeoff between global search which is often slow and a fast local search. Another advantage of this method is that we can use any algorithms we like at different stages of the search or even at different steps of iterations. This makes it easy to combine the advantages of various algorithms so as to produce better results. The main steps are outlined in Fig. 1.

Here the only parameter is p_e which controls the switch between local and global search. That is, it controls when to do exploitation and when to do exploration. It is worth pointing out that this is a methodology or strategy, not an algorithm. In fact, we can use different algorithms at different stages and at different time of the iterations. The algorithm used for the global exploration should have enough randomness so as to explore the search space diversely and effectively. This process is typically slow initially, and should speed up as the system converges (or no better solutions can be found after a certain number of iterations). On the other hand, the algorithm used for the intensive local exploitation should be an efficient local optimizer. The idea is to reach the local optimality as quickly as possible, with the minimal number of function evaluations. This stage should be fast and efficient.

```
Objective functions $f(\boldsymbol{x})$
Initialization and random initial guess $\boldsymbol{x}^{t=0}$
while (stop criterion)
    Global exploration by randomization
    Evaluate the objectives and find a promising solution
    If $p_e$ <rand, switch to a local search
    Intensive local search around a promising solution
    via an efficient optimizer
        if (a better solution is found)
        Update the current best
        end
    end
    Update $t = t + 1$
end
Post-process the results and visualization.
```

Fig. 1. Pseudo code of the eagle strategy.

A. Flower Pollination Algorithm

Flower pollination algorithm (FPA), or flower algorithm, was developed by Xin-She Yang in 2012 [17], inspired by the flow pollination process of flowering plants. For simplicity, flower algorithm uses the following four rules:

1) Biotic and cross-pollination can be considered as a process of global pollination process, and pollen-carrying pollinators move in a way which obeys Lévy flights (Rule 1).
2) For locall pollination, abiotic and self-pollination can be used (Rule 2).
3) Pollinators such as insects can develop flower constancy, which is equivalent to a reproduction probability that is proportional to the similarity of two flowers involved (Rule 3).
4) The interaction or switching of local pollination and global pollination can be controlled by a switch probability $p \in [0, 1]$, with a slight bias towards local pollination (Rule 4).

In order to formulate updating formulae, we have to convert the above rules into updating equations. For example, in the global pollination step, flower pollen gametes are carried by pollinators such as insects, and pollen can travel over a long distance because insects can often fly and move in a much longer range. Therefore, Rule 1 and flower constancy can be represented mathematically as

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \gamma L(\lambda)(\boldsymbol{g}_* - \boldsymbol{x}_i^t), \tag{4}$$

where $\boldsymbol{x}_i^t$ is the pollen i or solution vector $\boldsymbol{x}_i$ at iteration t, and $\boldsymbol{g}_*$ is the current best solution found among all solutions at the current generation/iteration. Here γ is the parameter that corresponds to the strength of the pollination, which essentially is also a step size. Since insects may move over a long distance with various distance steps, we can use a Lévy flight to mimic this characteristic efficiently. That is, we draw L from a Levy distribution

$$L \sim \frac{\lambda\Gamma(\lambda)\sin(\pi\lambda/2)}{\pi}\frac{1}{s^{1+\lambda}}, \quad (s \gg s_0 > 0). \tag{5}$$

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

For the local pollination, both Rule 2 and Rule 3 can be represented as

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \epsilon(\boldsymbol{x}_j^t - \boldsymbol{x}_k^t), \tag{6}$$

where $\boldsymbol{x}_j^t$ and $\boldsymbol{x}_k^t$ are pollen from different flowers of the same plant species. This essentially mimics the flower constancy in a limited neighborhood. Mathematically, if $\boldsymbol{x}_j^t$ and $\boldsymbol{x}_k^t$ comes from the same species or selected from the same population, this equivalently becomes a local random walk if we draw ϵ from a uniform distribution in [0,1].

In principle, flower pollination activities can occur at all scales, both local and global. But in reality, adjacent flower patches or flowers in the not-so-far-away neighborhood are more likely to be pollinated by local flower pollen than those far away. In order to mimic this feature, we can effectively use a switch probability (Rule 4) or proximity probability p to switch between common global pollination to intensive local pollination. To start with, we can use a naive value of $p = 0.5$ as an initially value. A parametric study showed that $p = 0.8$ may work better for most applications.

Preliminary studies suggest that flower algorithm is very efficient, and has been extended to multi-objective optimization [24].

B. ES with FPA

As ES is a two-stage strategy, we can use different algorithms at different stage. The large-scale coarse search stage can use randomization via Lévy flights. In the context of metaheuristics, the so-called Lévy distribution is a distribution of the sum of n identically and independently distribution random variables.

For the second stage, we can use differential evolution as the intensive local search. We know FPA is a global search algorithm, it can easily be tuned to do efficient local search by limiting new solutions locally around the most promising region. This can be easily achieved by setting γ to a very small value. Such a combination may produce better results than those by using pure FPA only, as we will demonstrate this later. Obviously, the balance of local search (intensification) and global search (diversification) is very important, and so is the balance of the first stage and second stage in the ES.

C. More Exploration?

All metaheuristic algorithms have algorithm-dependent parameters. For flower algorithm, parametric studies suggest that we can use $n = 15$, $p = 0.8$, $\gamma = 0.1$ and $\lambda = 1.5$ for the applications here. In terms of the balance of exploration and exploitation in the combination of eagle strategy and flower algorithm, the most important parameter is p_e in the eagle strategy.

For isotropic random walks for local exploration, we have $D \approx s^2/2$ where s is the step length with a jump during a unit time interval or each iteration step. From equation (3), the optimal ratio of exploitation and exploration in a special case of $R/a \gg 1$, and we have

$$\tau_a/\tau_R^2 \approx 1/8. \tag{7}$$

This implies that more times should spend on the exploration stage. It is worth pointing out that the naive guess of 50-50 probability in each stage is not the best choice. More efforts should focus on the exploration so that the best solutions found by the algorithm can be globally optimal with possibly the least computing effort.

In the case studies to be described below, we have used the eagle strategy with combination with cuckoo search to find the optimal solutions to two design benchmarks and we found that the optimal ratio is between 0.15 to 0.25, which are roughly close to the above theoretical result. This may imply that ES with FPA has an intrinsic ability of balancing exploration and exploitation close to true optimal.

IV. Numerical Experiments and Design Benchmarks

There are a wide range of design benchmarks, and it is not possible to use even a good fraction of these benchmarks in a short paper. Therefore, we will use two well-selected case studies to demonstrate how metaheuristic algorithms perform for nonlinear global optimization problems.

A. Standing-Wave Function

Let us first use a multimodal test function to see how to find the fine balance between exploration and exploitation in an algorithm for a given task. A standing-wave test function, formulated by Xin-She Yang [15], can be a good example

$$f(\boldsymbol{x}) = 1 + \Big\{ \exp[-\sum_{i=1}^{d} (\frac{x_i}{\beta})^{10}] \\ -2\exp[-\sum_{i=1}^{d} x_i^2] \Big\} \cdot \prod_{i=1}^{d} \cos^2 x_i, \tag{8}$$

which is multimodal with many local peaks and valleys. It has a unique global minimum at $f_{\min} = 0$ at $(0, 0, ..., 0)$ in the domain $-20 \le x_i \le 20$ where $i = 1, 2, ..., d$ and $\beta = 15$. In this case, we can estimate that $R = 20$ and $a \approx \pi/2$, this means that $R/a \approx 12.7$, and we have in the case of $d = 2$

$$p_e \approx \tau_{\text{optimal}} \approx \frac{1}{2[2 - 1/\ln(R/a)]^2} \approx 0.19. \tag{9}$$

This indicate that the algorithm should spend 80% of its computational effort on global explorative search, and 20% of its effort on local intensive search.

For the eagle strategy with flower pollination algorithm (ES with FPA), we have used $n = 15$ and 1000 iterations. We have varied the switching probability p_e which essentially controls the exploration and exploitation in the strategy, and p_e can thus affect the solution quality. A set of 40 numerical experiments have been carried out for each value of p_e and the results are summarized in Table 1.

TABLE I
Variations of p_e and its effect on the solution quality.

p_e	0.4	0.3	0.2	0.1	0.05	0.025
$f_{\min}$	9.3E-11	1.1E-12	2.3E-14	6.8E-12	8.5E-11	4.7E-9

This table clearly shows that $p_e \approx 0.2$ provides the optimal balance of local exploitation and global exploration, which is consistent with the theoretical estimation. Though there is no direct analytical results for higher dimensions, we can expect that more emphasis on global exploration is also true for higher dimensional optimization problems. Let us look a benchmark with a few design variables.

B. Welded Beam Design

Beam design problems are widely used benchmarks for engineering optimization. For example, the standard welded beam design problem has four design variables: the width w and length L of the welded area, the depth h and thickness h of the main beam. The objective is to minimise the overall fabrication cost, under the appropriate constraints of shear stress τ, bending stress σ, buckling load P and maximum end deflection δ [4], [23]. The problem can be written as

$$\text{minimise} \quad f(\boldsymbol{x}) = 1.10471 w^2 L + 0.04811 dh(14.0 + L), \tag{10}$$

subject to

$$\begin{aligned} g_1(\boldsymbol{x}) &= w - h \le 0, \\ g_2(\boldsymbol{x}) &= \delta(\boldsymbol{x}) - 0.25 \le 0, \\ g_3(\boldsymbol{x}) &= \tau(\boldsymbol{x}) - 13,600 \le 0, \\ g_4(\boldsymbol{x}) &= \sigma(\boldsymbol{x}) - 30,000 \le 0, \\ g_5(\boldsymbol{x}) &= 0.10471 w^2 + 0.04811 hd(14 + L) - 5.0 \le 0, \\ g_6(\boldsymbol{x}) &= 0.125 - w \le 0, \\ g_7(\boldsymbol{x}) &= 6000 - P(\boldsymbol{x}) \le 0, \end{aligned} \tag{11}$$

where

$$\begin{aligned} &\sigma(\boldsymbol{x}) = \tfrac{504,000}{hd^2}, \quad Q = 6000(14 + \tfrac{L}{2}), \\ &D = \tfrac{1}{2}\sqrt{L^2 + (w + d)^2}, \quad J = \sqrt{2}\, wL[\tfrac{L^2}{6} + \tfrac{(w+d)^2}{2}], \\ &\delta = \tfrac{65,856}{30,000 hd^3}, \quad \beta = \tfrac{QD}{J}, \\ &\alpha = \tfrac{6000}{\sqrt{2}wL}, \quad \tau(\boldsymbol{x}) = \sqrt{\alpha^2 + \tfrac{\alpha\beta L}{D} + \beta^2}, \\ &P = 0.61423 \times 10^6 \ \tfrac{dh^3}{6}(1 - \tfrac{d\sqrt{30/48}}{28}). \end{aligned} \tag{12}$$

The simple limits or bounds are $0.1 \le L, d \le 10$ and $0.1 \le w, h \le 2.0$. For example, using both ES with FPA, we have obtained the same as those in the literature [4]

$$f_* = 1.724852 \tag{13}$$

at

$$(0.205730, 3.470489, 9.036624, 0.205729). \quad (14)$$

However, ES with FPA uses far fewer function evaluations. In fact, our simulation only uses about 10% of the computational effort, compared with other methods reported in the literature.

V. Discussion and Conclusions

Nature-inspired metaheuristic algorithms have recently gained increasing popularity. Metaheuristic algorithms have the flexibility and ability of dealing with nonlinear global optimization problems. The current work has highlighted the importance of key components of exploration and exploitation in metaheuristics. By using examples, we estimated the ratio of search times or efforts of exploitation and exploration stages, which was based on intermittent search theory in the 2D case. Though the intermittent search theory can provide unique insight into the unknown landscape of multimodal problems by placing more emphasis on the global exploration, it does not provide generic answers to all problems. In fact, optimality in this case is landscape-based, because it is relevant to a specific type of problem while assuming the optimization algorithm used does not incorporate any knowledge of the problem of interest. In reality, any algorithms that can effectively use such problem-specific knowledge will be more efficient than those that do not.

Obviously, how to incorporate such knowledge into an algorithm is still an unresolved problem, despite the studies and progress in machine learning, data mining and computational intelligence. Therefore, a good research topic would be the ways of incorporating knowledge into optimization algorithms to enhance search efficiency. Further research can focus on the case studies in higher dimensions. It may also be very fruitful to investigate various type of optimization problems and see how the distribution of modes and targets can be associated with the performance of an algorithm, which may help to identify the best algorithms for each type of problem.

References

[1] Bénichou, O., Loverdo, C., Moreau, M., and Voituriez, R., Two-dimensional intermittent search processes: An alternative to Lévy flight strategies, *Phys. Rev.*, E**74**, 020102(R), (2006).

[2] Bénichou, O., Loverdo, C., Moreau, M., and Voituriez, R., Intermittent search strategies, *Review of Modern Physics*, **83**, 81-129 (2011).

[3] Blum, C. and Roli, A. (2003) Metaheuristics in combinatorial optimisation: Overview and conceptural comparision, *ACM Comput. Surv.*, Vol. 35, 268-308.

[4] Cagnina, L. C., Esquivel, S. C., and Coello, C. A., Solving engineering optimization problems with the simple constrained particle swarm optimizer, *Informatica*, **32**, 319-326 (2008).

[5] B. J. Copeland, *The Essential Turing*, Oxford University Press, 2004.

[6] Corne D. and Knowles, J., Some multiobjective optimizers are better than others, *Evolutionary Computation*, CEC'03, **4**, 2506-2512 (2003).

[7] Dorigo, M. and Stütle, T., *Ant Colony Optimization*, MIT Press, (2004).

[8] Floudas, C. A. and Pardolos, P. M., *Encyclopedia of Optimization*, 2nd Edition, Springer (2009).

[9] Gandomi, A. H., Yang, X. S., Talatahari, S., and Deb, S., (2012). Coupled eagle strategy and differential evolution for unconstrained and constrained global optimization, *Computers & Mathematics with Applications*, **63**(1), 191-200 (2012).

[10] Gandomi, A. H., Yun, G. J., Yang, X. S., Talatahari, S., (2013). Chaos-enhanced accelerated particle swarm optimization, *Communications in Nonlinear Science and Numerical Simulation*, Vol. 18, No. 2, pp. 327-340.

[11] Kennedy, J. and Eberhart, R. (1995) Particle swarm optimisation, in: *Proc. of the IEEE Int. Conf. on Neural Networks*, Piscataway, NJ, pp. 1942-1948.

[12] Parpinelli, R. S., and Lopes, H. S., New inspirations in swarm intelligence: a survey, *Int. J. Bio-Inspired Computation*, **3**, 1-16 (2011).

[13] Shlesinger, M. F., Random searching, *J. Phys. A.: Math. Theor.*, **42**, 434001 (2009).

[14] X. S. Yang, *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, UK, 2008

[15] Yang, X. S. (2010), *Engineering Optimization: An Introduction with Metaheuristic Applications*, John Wiley and Sons, USA (2010).

[16] Yang, X. S., (2011). Review of meta-heuristics and generalised evolutionary walk algorithm, *Int. J. Bio-Inspired Computation*, Vol. 3, No. 2, pp. 77-84.

[17] Yang, X. S. (2012), Flower pollination algorithm for global optimization, in: *Unconventional Computation and Natural Computation*, Lecture Notes in Computer Science, Vol. 7445, pp. 240-249.

[18] Yang, X. S., (2013). Multiobjective firefly algorithm for continuous optimization, *Engineering with Computers*, Vol. 29, No. 2, pp. 175-184.

[19] Yang, X. S. and Deb, S., (2010). Eagle strategy using Lévy walk and firefly algorithm for stochastic optimization, in: *Nature Inspired Cooperative Strategies for Optimization* (NICSO 2010) (Eds. J. R. Gonzalez et al.), SCI Vol. 284, 101-111.

[20] Yang, X. S. and Koziel, S., (2011). *Computational Optimization and Applications in Engineering and Industry*, Studies in Computational Intelligence, Vol. 359, Springer, Heidelberg.

[21] Yang, X. S. and Deb, S., (2012). Two-stage eagle strategy with differential evolution, *Int. J. Bio-Inspired Computation*, **4**(1), 1-5 (2012).

[22] Yang, X. S., and Deb, S., (2013). Multiobjective cuckoo search for design optimization, *Computers & Operations Research*, Vol. 40, No. 6, pp. 1616-1624.

[23] Yang, X. S. and Gandomi, A. H., Bat algorithm: a novel approach for global engineering optimization, *Engineering Computations*, **29**(5), pp. 464-483 (2012).

[24] Yang, X. S., Karamanoglu, M., He, X. S., (2013). Multiobjective flower algorithm for optimization, *Procedia Computer Science*, Vol. 18, pp. 861-868.