



# Exploiting flower constancy in flower pollination algorithm: improved biotic flower pollination algorithm and its experimental evaluation

Paweł Kopciwicz<sup>1</sup> · Szymon Łukasik<sup>1,2</sup>

Received: 20 November 2018 / Accepted: 29 March 2019  
© The Author(s) 2019

## Abstract

Recent growth of metaheuristic search strategies has brought a huge progress in the domain of computational optimization. The breakthrough started since the well-known Particle Swarm Optimization algorithm had been introduced and examined. Optimization technique presented in this contribution mimics the process of flower pollination. It is build on the foundation of the first technique of this kind—known as Flower Pollination Algorithm (FPA). In this paper, its simplified and improved version, obtained after extensive performance testing, is presented. It is based on only one natural phenomena—called flower constancy—the natural mechanism allowing pollen carrying insects to remember the positions of the best pollen sources. Modified FPA, named as Biotic Flower Pollination Algorithm (BFPA) and relying solely on biotic pollinators, outperforms original FPA, which itself proved to be very effective approach. The paper first presents a short description of original FPA and the changes leading to Biotic Flower Pollination Algorithm. It also discusses performance of the modified algorithm on a full set of CEC17 benchmark functions. Furthermore, in that aspect, the comparison between BFPA and other optimization algorithms is also given. Finally, brief exemplary application of modified algorithm in the field of probabilistic modeling, related to physics and engineering, is also presented.

**Keywords** Metaheuristics · Optimization · Nature-inspired algorithms · Flower pollination algorithm

## 1 Introduction

Nature-inspired metaheuristics constitute an important element of Computational Intelligence revolution, due to their high performance and reasonable quality of obtained results. Recent years brought intensive growth of this field. A variety of new methods have been introduced, which—while being criticized for lack of methodological novelty and relying only on attractive metaphors [30]—were often demonstrated to perform very well for benchmark optimization problems. The inspiration for these algorithms typically comes from one of four types of natural phenomena: evolution, swarm behavior, physical processes or

human activity. Common example of the first class is the Genetic Algorithm [29]; however, more recent Differential Evolution technique [31] can also be named in this context. Swarming behaviors exhibited by a variety of living organisms serve as a foundation for a large number of techniques such as: Grey Wolf Optimizer [19], Grasshopper Optimization Algorithm [25], Krill Herd Algorithm [12] or standard Particle Swarm Optimization [9] to name just a few. As typical representatives of the third class—techniques inspired by physical processes—along the most commonly known Simulated Annealing algorithm [15] more recent methods of Gravitational Search Algorithm [23] or Sine Cosine Algorithm [18] can be presented. Finally, analyzing human behavior and activities lead to the creation of, inter alia, Fireworks Algorithm [32] and Brainstorming [27]. For a more extensive overview of nature-inspired optimization algorithms, one could refer to a recent extensive survey [11].

The natural phenomena, which became the core of algorithm presented in this paper, is one of the flower pollination processes, more accurately—the flower

---

✉ Szymon Łukasik  
slukasik@ibspan.waw.pl

<sup>1</sup> Faculty of Physics and Applied Computer Science, AGH University of Science and Technology, al. Mickiewicza 30, 30-059 Kraków, Poland

<sup>2</sup> Systems Research Institute, Polish Academy of Sciences, ul. Newelska 6, 01-447 Warsaw, Poland

constancy mechanism. Flower constancy [8] allows the insects, which are able to carry a pollen (e.g., honey bees), to remember the paths between best-known pollen sources. The original optimization technique based on flower pollination, called Flower Pollination Algorithm (FPA), comes from 2012 and has been introduced by Yang [37]. This algorithm imitated two different variants of flower pollination and classified them into local pollination—where all actions occurred locally, and global pollination—where the pollen was moved to partly random, often distant location [20]. With more insightful approach, local pollination could be compared to flower constancy, being responsible for pollen carriers attraction to advantageous flower types, which were positively scrutinized in the past. Thus, local pollination imitated carriers communication and tendency to remain at lucrative habits. In practice, conjunction of this mechanism and randomized global pollination led to creation of interesting optimization method, which quickly explores the given function landscape to get to its optimal domain, simultaneously preventing the algorithm from getting stuck in local extremes. Since the foundation of the original FPA, many researchers have studied it with different results [1, 17, 34], and also modified standard algorithm's structure [26, 28], but the basic idea behind FPA has remained unchanged. The goal of this article is to present improved and simplified FPA version, which has been created to increase algorithm performance, and to reduce computational cost and total time of algorithm's execution. It is not a new swarm intelligence technique but a modification of the existing algorithm inspired by experimental studies on its behavior. This work is motivated by the fact that in the most recent comparative experimental studies [2, 4], the FPA proved to be highly competitive—even in reference to the more recent algorithms. It has also been successfully adopted in many engineering and scientific applications [21].

This paper is constructed as follows. Firstly, the natural process of pollination—which has become an archetype of the algorithm—is described. Then, the structure of the algorithm itself, as well as the changes introduced by authors of this paper are presented. Subsequently, the performance of modified algorithm with regard to other known algorithms (PSO, original FPA, slightly modified FPA with improved random factor—LEFPA [28] and Symbiotic Organism Search (SOS) [7]) is demonstrated. It is followed by use-case example of application of the proposed algorithm. Finally, further possible improvements are communicated.

## 2 Flower pollination metaphor in optimization

Plant pollination in nature can occur as a consequence of several different mechanisms. First of these is self-pollination, where the pollen comes from the same flower or different flower of the same plant. In contrast to that, cross-pollination takes place when the pollen comes from different plant [14]. In original FPA, inspired by pollination, all flowers were assumed to be of the same type [37], so all events could be classified into self-pollination category.

Another classification type divides pollination processes into two groups, biotic and abiotic. Biotic pollination occurs while insects, also called pollinators, carry pollen between consecutive flowers. Abiotic pollination in turn is done through the presence of independent and random factors like wind or water. It was established through biological research that over 80% of all pollinations occurring in nature are in fact biotic [3]. Abiotic pollination can be considered as the one introducing more randomness to the process of pollen transfer and, as we suggest later, this behavior could be neglected in building algorithm structure.

The algorithm inspired by pollination introduces third categorization of pollination process—based on its range: local or global. It is worth to mention that original FPA employs both of them. The first—local pollination theoretically could be considered as either biotic or abiotic, because the flowers are in this case located close to each other. It means that both natural factors such as wind and pollinator activities could be perceived as of significant importance. Still local pollination used in the original FPA technique was limited to be only biotic and represents the flower constancy, mentioned in the previous section. It is based on random walks between flowers. Global pollination, which is a second part of the algorithm, is the long-distance pollination. In this case, FPA focuses mainly on abiotic pollination as successful long-range wind pollination can also be observed in Nature, e.g., for open areas where pollen can be carried far without any obstruction. In FPA, the trajectories of pollen displacement are in this case being rolled using Levy distribution [22]. Angular part of Levy distribution is symmetric; thus, direction of displacement must be simultaneously rolled from uniform distribution. It means that simulation of abiotic global pollination modeled by FPA brings more randomness and unpredictability to the pollination process.

Optimization deployment of described processes is based on locating flowers on functional landscape. The function value reflecting flower position determines the amount of pollen produced by this flower. Detailed

description of algorithms based on aforementioned natural processes is provided in the following sections.

### 3 Specification of original FPA and existing modifications

The population in Flower Pollination Algorithm consists of  $N$  flowers of the same type. Each flower is denoted a vector  $x_n$ ,  $n = 1, \dots, N$  and represents exactly one solution in the domain of tested function  $f$ . The goal of the optimization is to find a value of  $x_n$ —within the feasible search space  $S \subset R^D$ —denoted as  $x^*$  such as  $x^* = \operatorname{argmin}_{x \in S} f(x)$ , assuming that the goal is to minimize cost function  $f$ . Population in FPA is at first initialized randomly. After initialization, in every step of algorithm's main iteration  $k$ , all flowers in sequence are selected to be pollinated. It is controlled by parameter  $p$ , named as switch probability, which determines the probability of global pollination. If a flower is chosen not to be pollinated this way, a local pollination is executed instead. The whole process is repeated iteratively until predetermined termination criterion is found to be fulfilled.

Let us denote  $x_n(k)$  as solution  $n$  in iteration  $k$ ,  $x^*(k)$ —as the best solution found so far and  $s_0, \gamma$  as global pollination parameters. Furthermore, let  $x_{n,\text{trial}}$  to represent a new candidate solution obtained through the pollination phase. It becomes a new location of solution  $n$  if its quality—in terms of cost function  $f$ —is better than the quality of the original one. Algorithm 1 provides the description of pollination process in FPA using this notation.

---

#### Algorithm 1 Flower Pollination Algorithm - pollination schemes

---

```

1: if Real_Rand_in_(0, 1) <  $p$  then
2:   {Global pollination}
3:    $s \leftarrow \text{Levy}(s_0, \gamma)$ 
4:    $x_{n,\text{trial}} \leftarrow x_n(k) + s(x^*(k) - x_n(k))$ 
5: else
6:   {Local pollination}
7:    $\epsilon \leftarrow \text{Real_Rand_in}$ _(0, 1)
8:    $r, q \leftarrow \text{Integer_Rand_in}$ (1,  $N$ )
9:    $x_{n,\text{trial}} \leftarrow x_n(k) + \epsilon(x_q(k) - x_r(k))$ 
10: end if

```

---

Flower Pollination Algorithm as an effective optimizer found variety of modifications—enhancing its capabilities

and improving performance. Besides, more specific FPA variants for combinatorial [33] and binary optimization [10] more generic changes—related to the structure of the algorithm have been proposed [26]. It includes newly introduced Linear–Exponential Flower Pollination Algorithm [28] based on customized switching of probability  $p$ . In addition to that adding a chaotic component in a form of chaotic maps [13] was also considered—it is one of the popular approaches used recently in heuristic optimization. An improvement in new mutation operators and modified local search was also under investigation [24]. A list of modifications also includes new population initialization methods [36]. For a more extensive overview of changes proposed for the standard FPA, one could refer to recent state-of-art reviews [5, 21].

### 4 Proposed approach

As already demonstrated in original FPA, the probability of local and global pollination for single flower is defined by the probability factor  $p$ , for which the optimal value of 0.8 was suggested, according to the contributions of algorithm's developer Yang [37] and other sources [17]. However, our intensive experimentation exhibited that in most cases FPA produces the best results while the value of  $p$  drifts to 0—in other words, when the system tends to eliminate completely the global pollination process. As it was presented in Sect. 2, random walks have the strict nature of randomness that interferes the methodical, intuitive behavior of pollinators. Removing global pollination process also simplifies the scheme of FPA, because rolling random numbers from Levy distribution (which has the infinite variance) might be problematic and it slightly extends the algorithm execution time. In addition to that, in this way an elimination of two algorithm's parameters:  $s_0$  and  $\gamma$  is achieved. Computational cost of algorithm remains at  $O(k, N)$ , where  $k$  is iteration number,  $N$  is the size of flower population. By removing global pollination and forcing the flowers to only pollinate by flower constancy mechanism, simplified variant of FPA is created, which can be naturally called Biotic Flower Pollination Algorithm (BFPA). The complete structure of BFPA is presented as Algorithm 2.

**Algorithm 2** Biotic Flower Pollination Algorithm

---

```

1: Define objective function  $f(x)$ , where  $x = [x_1, x_2, \dots, x_D]$ 
2: Generate  $N$  random solutions (initial positions for  $N$  flowers)
3: Evaluate initial solutions
4: for  $k = 1 : IterationNumber$  do
5:   for  $i = 1 : N$  do
6:     Draw  $\epsilon$  from uniform distribution  $[-C, C]$ , where  $C$  is constant
7:     Randomly choose flowers  $p$  and  $q$  from the population
8:     Perform pollination:
9:      $x_i(k+1) = x_i(k) + \epsilon(x_p(k) - x_q(k))$ 
10:    Evaluate new solution
11:    Check the bounds: If  $x_i(k+1) \notin S$ , generate random  $x_i(k+1)$  within  $S$ 
12:    If  $x_i(k+1)$  is better  $x_i(k)$ , update position in the population
13:   end for
14: end for

```

---

Another modification taken into consideration was the factor of local pollination  $\epsilon$ . Adding the sign of its value does not change the structure of the algorithm—and at the same time, it causes the pollination steps to be extended. In practice, modifying  $\epsilon$  value heavily influences received results, and it can be helpful if we want to obtain the best possible solution for given problem. Through our experimentation—selection of which will be enclosed in subsequent part of the paper—we established that as the best universal value  $C$  equal to 1 can be used.

The results of BFPA optimization performed on benchmark continuous problems, with regard to other exemplary and indicative algorithms are presented in the next section.

## 5 Experimental performance

The source of all benchmark functions used in performance tests was the CEC17 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization [6]. It contains a variety of different black-box continuous benchmark problems. A short summary of full suite is presented in Table 1, and its detailed specification can be found in [6]. First nine functions are shifted and rotated common and well-known optimization problems, from the simplest with only one local minimum, e.g., Bent Cigar function (F1), to more complex with extremely high number of local minima, e.g., Schaffer function (F6). Functions of number 10–20 are hybrids, that are the linear combination of several basic functions (mostly of these already used in a basic set F1–F9). The last 10 functions are compositions, that in turn represent couplings between several previous functions in different frames of optimization domain.

All functions are assumed to be 10-dimensional and stretched on domain  $S \in [-100, 100]^{10}$ . During

**Table 1** Continuous benchmark function suite used to examine the performance of BFPA

Function	Type of function
F1	Shifted and Rotated Bent Cigar Function
F2	Shifted and Rotated Zakharov Function
F3	Shifted and Rotated Rosenbrock's Function
F4	Shifted and Rotated Rastrigin's Function
F5	Shifted and Rotated Expanded Schaffer's Function
F6	Shifted and Rotated Lunacek Bi-Rastrigin Function
F7	Shifted and Rotated Non-continuous Rastrigin's Function
F8	Shifted and Rotated Levy Function
F9	Shifted and Rotated Schwefel's Function
F10–F19	Hybrid Functions
F20–F29	Composition Functions

All functions come from CEC17 competition [6]

performance tests, we assumed that number of evaluation of single function is limited to 100,000, as the CEC17 competition rules prescribe. This pool was divided into 2850 iterations of the main algorithm loop and the population of 35 agents. It is equivalent to 99750 evaluations of a specific black-box problem, which is within the limits of competition rules.

As the alternative optimization algorithms—used to evaluate the performance of BFPA—besides FPA (with  $p = 0.8$ ), standard Particle Swarm Optimization, state-of-art FPA variant—linear-exponential Flower Pollination Algorithm (LEFPA), and Symbiotic Organism Search (SOS) were used. For PSO, as the  $C1$  and  $C2$  parameters typical values of  $C1 = C2 = 2$  were employed [9]. For LEFPA  $p_{\text{initial}}$  and  $p_{\text{final}}$  inner parameters, values 0.8 and 0.6 were used, respectively, as the creators of LEFPA recommend [28]. SOS algorithm used in our experiments employ default settings as described in [7].

**Table 2** Algorithms' performance on CEC17 continuous benchmark problems

Funct.	Result	BFPA	FPA	LEFPA	PSO	SOS
F1	Mean	0	372.43	$1.79 \times 10^6$	$3.95 \times 10^3$	$2.2 \times 10^6$
	Min.	0	31.364	$2.47 \times 10^5$	1.153	$2.41 \times 10^3$
	SD	0	259.49	$1.27 \times 10^6$	$5.28 \times 10^3$	$6.78 \times 10^6$
F2	Mean	0	$6.07 \times 10^{-9}$	70.573	$4.17 \times 10^{-14}$	$2.84 \times 10^{-15}$
	Min.	0	$3.11 \times 10^{-10}$	10.613	0	0
	SD	0	$6.01 \times 10^{-9}$	72.500	$2.94 \times 10^{-14}$	$1.25 \times 10^{-14}$
F3	Mean	0	0.0016	5.702	3.193	0.0979
	Min.	0	$1.34 \times 10^{-7}$	3.104	0.2873	0.0259
	SD	0	0.0111	1.251	0.8638	0.0775
F4	Mean	5.947	14.286	13.010	7.918	6.938
	Min.	1.990	7.541	6.969	1.990	2.024
	SD	2.384	4.247	3.20	3.024	3.391
F5	Mean	0.2368	4.783	1.592	$2.37 \times 10^{-8}$	$6.63 \times 10^{-6}$
	Min.	$1.63 \times 10^{-5}$	1.750	0.547	0	$1.14 \times 10^{-13}$
	SD	0.6487	2.429	0.824	$1.83 \times 10^{-7}$	$1.32 \times 10^{-5}$
F6	Mean	15.950	26.229	30.920	18.870	27.544
	Min.	7.618	16.488	22.595	5.606	13.266
	SD	2.590	4.694	4.277	6.521	6.356
F7	Mean	5.747	14.082	11.486	10.085	7.266
	Min.	0.9950	6.502	4.641	3.980	1.041
	SD	2.325	3.845	2.692	3.682	2.936
F8	Mean	0	1.712	3.177	0.0015	0.0104
	Min.	0	0.0078	0.3159	0	0
	SD	0	1.194	5.818	0.0116	0.029
F9	Mean	663.02	650.11	626.57	287.07	174.91
	Min.	166.59	370.78	288.60	6.892	6.955
	SD	192.81	111.82	176.35	176.01	102.39
F10	Mean	0.1243	4.415	16.470	4.42	2.795
	Min.	0	1.445	6.165	0.0341	0.0256
	SD	0.3228	1.234	6.738	2.727	1.889
F11	Mean	8.380	$1.11 \times 10^3$	$4.78 \times 10^4$	$2.29 \times 10^4$	$1.48 \times 10^4$
	Min.	$8.98 \times 10^{-5}$	346.73	$8.52 \times 10^3$	336.29	261.11
	SD	18.329	323.73	$4.95 \times 10^4$	$2.90 \times 10^4$	$1.53 \times 10^4$
F12	Mean	4.052	14.620	$2.89 \times 10^3$	$5.49 \times 10^3$	$2.93 \times 10^3$
	Min.	$4.99 \times 10^{-5}$	6.331	490.13	14.424	7.756
	SD	1.959	4.423	$1.73 \times 10^3$	$6.74 \times 10^3$	$2.73 \times 10^3$
F13	Mean	0.9602	16.542	84.31	26.190	93.082
	Min.	0	7.754	41.260	2.051	0.3019
	SD	1.003	3.876	21.040	13.391	197.42
F14	Mean	0.2306	5.733	153.03	27.513	61.507
	Min.	0.0297	3.063	44.009	1.156	0.41
	SD	0.1886	1.781	74.263	24.817	134.62
F15	Mean	3.620	7.196	26.106	22.683	41.206
	Min.	1.249	1.874	4.3246	0.0448	0.2698
	SD	1.859	6.040	17.15	47.860	57.057
F16	Mean	32.729	41.923	44.133	18.617	8.283
	Min.	25.704	25.887	27.993	0.3122	0.0197
	SD	4.217	7.363	7.122	22.526	8.513

Table 2 (continued)

Funct.	Result	BFPA	FPA	LEFPA	PSO	SOS
F17	Mean	0.4570	42.151	$1.34 \times 10^3$	$4.74 \times 10^3$	$4.46 \times 10^3$
	Min.	0.3948	26.738	138.96	115.53	14.374
	SD	0.0378	9.689	$1.28 \times 10^3$	$5.12 \times 10^3$	$5.98 \times 10^3$
F18	Mean	1.106	5.182	543.49	24.041	289.09
	Min.	0.6459	2.959	51.059	0.5999	0.1049
	SD	0.3211	0.8035	548.75	25.499	654.72
F19	Mean	24.120	45.663	38.398	11.470	1.415
	Min.	0.9766	27.038	22.289	0	0
	SD	7.560	9.221	7.511	11.340	1.809
F20	Mean	141.58	111.75	119.52	195.16	106.68
	Min.	100	100	101.09	100	100
	SD	56.941	36.170	37.351	40.045	25.069
F21	Mean	80.215	80.117	71.606	104.62	98.975
	Min.	0	4.292	6.041	19.224	11.563
	SD	44.842	38.006	40.578	23.326	14.032
F22	Mean	308.21	319.46	313.53	309.34	310.28
	Min.	306.96	312.16	303.74	300	304.37
	SD	1.167	5.383	3.643	4.019	3.758
F23	Mean	288.29	205.53	306.22	311.46	276.7
	Min.	100	100.00	110.34	100	100
	SD	105.28	119.11	69.154	77.622	102.57
F24	Mean	397.86	399.44	412.77	414.65	421.19
	Min.	397.74	397.75	398.87	397.89	397.74
	SD	0.1633	8.479	15.716	23.073	23.095
F25	Mean	271.67	258.53	314.26	514.94	309.09
	Min.	0	0.0037	302.54	$4.55 \times 10^{-13}$	0
	SD	71.525	92.460	15.063	413.37	70.657
F26	Mean	371.30	389.21	395.98	394.91	391.83
	Min.	370.17	386.90	390.49	389.30	387.32
	SD	0.8716	0.7597	9.312	2.596	2.654
F27	Mean	549.74	300.00	418.29	447.21	386.81
	Min.	300.97	300	317.94	300	300
	SD	94.782	0.0017	121.71	161.02	130.12
F28	Mean	242.33	271.45	286.93	273.63	254.88
	Min.	229.47	241.61	248.84	234.91	236.07
	SD	7.372	18.699	23.274	41.955	13.581
F29	Mean	682.42	$3.13 \times 10^4$	$2.07 \times 10^5$	$4.28 \times 10^5$	$1.55 \times 10^5$
	Min.	249.47	467.79	$1.65 \times 10^3$	$1.44 \times 10^3$	723.32
	SD	631.80	$1.44 \times 10^5$	$2.86 \times 10^5$	$7.85 \times 10^5$	$3.17 \times 10^5$

For every benchmark problem, all algorithms were run 60 times and the average of optimization error, as well as its standard deviation and the best result from 60 consecutive runs were noted and are presented in Table 2. As an optimization error, the difference between final result and known global minimum value was chosen. To compare results, *t* Student Test for different variances, also known

as Welch Test [35], was used. Resulting value from Welch Test is *p* value, which represents probability of null hypothesis that expected values of two populations are equal. If *p* value is low, null hypothesis is rejected and both expected values; thus, both populations are considered as statistically different. Adoption of 0.95 confidence interval determines *p* value needs to be lower than 0.05 to

**Table 3** Overview of Welch test  $p$  value, which represent the probability of two selected samples of algorithm results being statistically insignificant

No.	BFPA-FPA	BFPA-LEFPA	BFPA-PSO	BFPA-SOS
F1	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$
F2	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p = 0.0836$
F3	eq, $p = 0.2687$	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$
F4	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p = 0.0001$	btr, $p = 0.0669$
F5	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	wrs, $p = 0.0064$	wrs, $p \leq 0.0001$
F6	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p = 0.0019$	btr, $p \leq 0.0001$
F7	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p = 0.0021$
F8	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	eq, $p = 0.3206$	btr, $p \leq 0.0001$
F9	eq, $p = 0.6548$	eq, $p = 0.2822$	wrs, $p \leq 0.0001$	wrs, $p \leq 0.0001$
F10	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$
F11	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$
F12	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$
F13	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$
F14	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$
F15	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p = 0.0031$	btr, $p \leq 0.0001$
F16	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	wrs, $p \leq 0.0001$	wrs, $p \leq 0.0001$
F17	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$
F18	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$
F19	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	wrs, $p \leq 0.0001$	wrs, $p \leq 0.0001$
F20	wrs, $p = 0.0009$	wrs, $p = 0.0137$	btr, $p \leq 0.0001$	wrs, $p \leq 0.0001$
F21	eq, $p = 0.9897$	eq, $p = 0.2724$	btr, $p = 0.0003$	btr, $p = 0.0028$
F22	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p = 0.0396$	btr, $p \leq 0.0001$
F23	wrs, $p \leq 0.0001$	eq, $p = 0.2728$	eq, $p = 0.1728$	eq, $p = 0.5425$
F24	eq, $p = 0.1535$	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$
F25	eq, $p = 0.3861$	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p = 0.0047$
F26	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$
F27	wrs, $p \leq 0.0001$	wrs, $p \leq 0.0001$	wrs, $p \leq 0.0001$	wrs, $p \leq 0.0001$
F28	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$
F29	eq, $p = 0.1055$	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$	btr, $p \leq 0.0001$

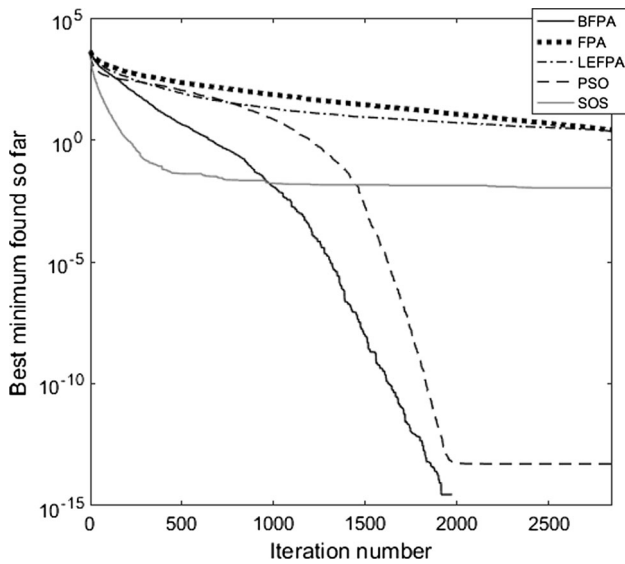
Table also presents which algorithm proved to be better, if  $p$  value was lesser than accepted significance level of 0.05

*btr* better, *eq* equal, *wrs* worse

recognize the results of two algorithms as different. After calculating  $p$  value for all cases, for majority of benchmark functions, we can state—even with probability 0.9999—that BFPA method gives significantly better results than other tested algorithms, which is shown in Table 3. It is worth to observe as well that the advantage of the proposed algorithm exhibits on all types of functions, their hybrids and compositions. Above all, for hybrid functions, BFPA was significantly better than any other variant of FPA for all considered benchmark instances. In addition to that, it was observed that FPA execution time was on average 20% longer than running time of BFPA.

In order to investigate algorithms' dynamics, the best minimum values found so far for all iterations were noted. Sample of 200 independent runs with the same initial population and random number generator seed for all algorithms were arranged, to avoid the effect of

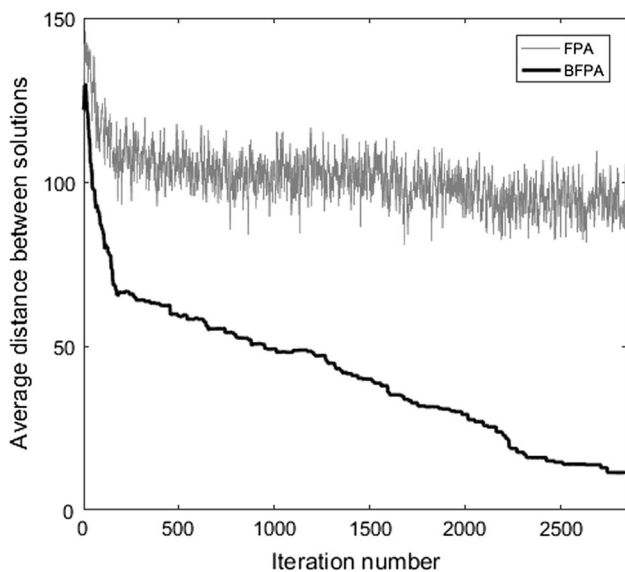
randomization. The average best solution of gradient comparison for all examined algorithms, calculated for F8 function, are presented in Fig. 1. Both FPA and LEFPA, which is FPA with improved global pollination process, acted similarly, with LEFPA converging a bit faster. SOS provided better results and was the leading algorithm over first 1000 iterations, however eventually was overtaken by PSO and BFPA. PSO algorithm most of times terminated reaching the actual position of global minimum. Finally, BFPA tended to global minimum quicker and more regularly, discovering desirable global minimum position every time. This result signifies that even though BFPA reduces random decisions during simulation, population does not remain in a local minima and always looks for the better position. This is most likely caused by removing usage of current best-known position in algorithm structure—that surprisingly often pulls the population into a local



**Fig. 1** Best minimum across optimization process for PSO, FPA, BFPA, SOS and LEFPA, calculated for F8 (Shifted and Rotated Levy Function) over 200 runs. This function has huge number of local minima [6]. Proposed algorithm—BFPA—found best position in every run, giving the best possible result in the shortest time. PSO found global minimum position as well, although not in every run, instead getting stuck several times in a local minimum

minimum trap. Similar experiments executed for selected CEC17 optimization problems with  $D = 30$  demonstrated that this positive feature is also observed for search spaces of higher dimensionality.

The differences between FPA and its modified version BFPA might be seen in diversity plots, presenting the average distance between solutions during optimization

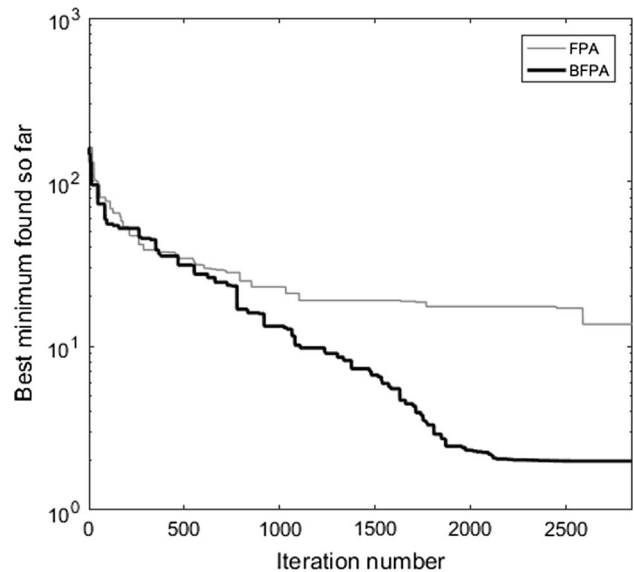


**Fig. 2** The average distance between solutions during algorithms’ run on benchmark function F4 (left plot). Due to random walk predominance over the pollen carries instinct, the exploitation ability of FPA

process for both FPA and BFPA in Fig. 2. The random walk contribution in FPA manifests in the form of the distance fluctuations. In the BFPA case, the exploration process is shortened and algorithm steps into desired area very quickly, continuing the exploitation phase with a linear decrease of the average distance between population agents. FPA keeps large distances between solutions over all iterations, which is the main reason of its worse exploitation abilities.

In order to examine the impact of the algorithm’s parameters on the optimizing performance, a number of different parameters set were chosen and examined in the same way as for the comparison of the algorithms, with 6 selected functions. All the results are collated in Table 4. The first three sets of parameters we have considered include different values taken for  $C$ —the gain of local pollination step—the middle, proposed value of  $C = 1$  and two marginal of  $C = 0.5$  and  $C = 1.5$ . At the same time, the population was fixed to  $N = 35$ . The results of this test are included in columns 3, 5 and 7 of the Table. The suggested value of  $C = 1$  gave the best result in 3 out of the other 6 cases; however, it was also the second best in the other 3 cases.

Another performance variability test concerned studying the effect of different number of flowers in the population with fixed value of  $C$ . A population lower than 18 flowers showed a tendency to stuck in one place, with an identical location vector. Since the full function suite was run for a population of 35 flowers, versions with 25 flowers and 50 flowers were added to the performance test, the results of which are also presented in Table 4. Once again, the



is limited, which is visualized in the form of best minimum found so far as a function of successive iterations (right plot). Both algorithms started from the same population



**Table 4** BFPA performance variability for different algorithm parameters

Funct.	Result	$C = 0.5, N = 35$	$C = 1, N = 25$	$C = 1, N = 35$	$C = 1, N = 50$	$C = 1.5, N = 35$
F1	Mean	0	$2.37 \times 10^{-16}$	0	$6.24 \times 10^{-12}$	0
	Min.	0	0	0	0	0
	SD	0	$1.83 \times 10^{-15}$	0	$1.74 \times 10^{-10}$	0
F4	Mean	15.71	8.971	5.947	8.02	6.748
	Min.	6.141	1.990	1.990	2.8738	1.990
	SD	4.492	4.078	2.384	2.617	2.646
F9	Mean	641.16	587.76	663.02	893.68	701.85
	Min.	292.60	125.72	166.59	511.83	247.96
	SD	157.01	185.58	192.81	171.89	221.70
F14	Mean	1.215	0.5828	0.2306	0.6376	0.3807
	Min.	0.3337	$1.01 \times 10^{-4}$	0.0297	0.2122	0.0061
	SD	0.6031	0.6068	0.1886	0.3146	0.4117
F19	Mean	53.157	17.996	24.120	37.513	19.154
	Min.	33.373	0.6244	0.9766	26.210	2.301
	SD	14.397	12.204	7.560	9.541	9.998
F24	Mean	333.01	398.30	397.86	351.63	402.42
	Min.	111.96	100	397.74	100	397.74
	SD	99.222	41.925	0.1633	107.22	13.917

proposed value of 35 flowers for 100, 000 function evaluations gave the best results, however the differences were not as significant, as for the different values of  $C$  provided. The most suitable number of flowers also depends on the dimensionality of a given problem. The  $C$  parameter, however, is independent of the number of dimensions, and if the initial population is randomly deployed over the function landscape, again its value of  $C = 1$  may be safely assumed.

### 6 Application example

BFPA’s simplicity allows it to be implemented in a wide range of applications. One of the functionalities we have already developed is fitting function models to the data, which follows combination of analytically defined distributions. If the function model consists of a huge number of independent variables, this task becomes impossible to complete swiftly through standard fitting techniques. Usage of BFPA is based on employing an objective function defined as of Mean Square Error (MSE) between data and current function fit, designated by a temporary set of model parameters of the best flower position vector. The algorithm minimizes objective function in relation to model parameters. These parameters are contingent on the model and might be simple, e.g., when fitting a Gaussian distribution or more complex when a mixture of different distributions is considered. The nature of MSE cost function, due to its high complexity and difference in contribution of parameters, can be compared to composition functions that

were part of CEC performance test. This means that fitting a function model to data points is often difficult and the global optimum cannot always be found. Nonetheless, it is a common task in data analysis and developing an algorithm which is simple and understandable is highly desirable.

As an example of application, fit to the model  $M(x)$  consisting of two independent Landau–Gauss distributions is presented (1):

$$M(x) = C_1(G \otimes L)(x_1) + C_2(G \otimes L)(x_2) + O(\zeta) \tag{1}$$

where  $(G \otimes L)(x)$  is a convolution of Gauss and Landau distribution (2):

$$(G \otimes L)(x) = \int_{-\infty}^{\infty} G(x')L(x - x')dx' \tag{2}$$

and  $G(x)$  and  $L(x)$  are (3), (4):

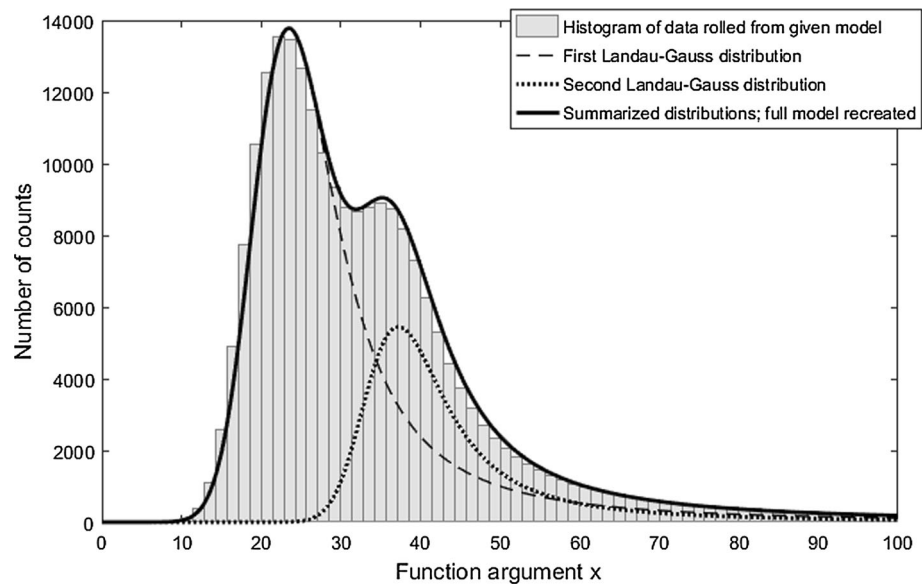
$$G(x) = A_1 \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right) \tag{3}$$

$$L(x) = A_2 \int_0^{\infty} \exp(-t) \cos\left(\frac{tx}{c} + 2\frac{t}{\pi} \ln\left(\frac{t}{c}\right)\right) dt \tag{4}$$

The Landau–Gauss convolution is then given by (5):

$$(G \otimes L)(x) = A \int_{-\infty}^{\infty} \int_0^{\infty} \exp(-t) \cos\left(\frac{t(x - x')}{c} + 2\frac{t}{\pi} \ln\left(\frac{t}{c}\right)\right) \exp\left(-\frac{(x' - \mu)^2}{2\sigma^2}\right) dt dx' \tag{5}$$

**Fig. 3** Exemplary result of BFPA optimization of function model parameters, based on Mean Square Error (LSE) minimization. The thick black line represents sum of both Landau–Gauss distributions, which are drawn separately. The histogram is constructed of 100 bins, grouping 25000 rolled numbers from a given model. Original and fitted parameters are, respectively, as follows:  
 $\mu_1 = 20$  (21.3),  
 $\sigma_1 = 2.24$  (2.23),  $c_1 = 5$  (4.98),  
 $A_1 = 3000$  (3044),  
 $\mu_2 = 36$  (36.9),  
 $\sigma_2 = 2.83$  (2.91),  $c_2 = 3$  (3),  
 $A_2 = 1000$  (982)



and  $O(\xi)$  is the numeric error of calculating a double improper integral within Landau–Gauss distribution, which is dependent on integrating technique and has a random character.

Since Landau distribution describes energy loss of the charged particle in a material, this model is often used in many fields of technology, starting with electronics, where it describes the energy loss of ionizing particles in silicon, ending with nuclear and high-energy physics, where particles with high momentum cross the detector material and lose energy. Convolution with Gauss distribution is not necessary when only energy loss prediction is the subject of consideration, yet it is essential if deposited energy is measured in non-zero time. To make the optimization task more challenging, another Landau–Gauss distribution was added to the model. As consequence of double improper integral in Landau–Gauss distribution, calculation cost of each function evaluation is significant. Hence, number of cost function evaluations should be as low as possible to avoid a long lasting optimization process. In the proposed model, the following parameters take part in optimization:

- $\mu$ —mean of the Gauss distribution
- $\sigma$ —standard deviation of Gauss distribution
- $c$ —Landau width parameter
- $A$ —amplitude of Landau–Gauss distribution

for each of Landau–Gauss distribution; thus, in summary we have 8 parameters that lead to 8 argument MSE-based cost functions.

The BFPA algorithm achieved a final result after an average of 422 iterations. The original FPA has also been tested and it needed more than 750 iterations; however, the

final result was identical in both attempts. The result of the model fitting using BFPA was demonstrated in Fig. 3.

## 7 Summary

The proposed modified variant of the Flower Pollination Algorithm has exhibited excellent results in the course of the experimental test procedure covered in previous sections, with both high-quality results and a decent time of execution being identified. BFPA proved to perform much better than the original FPA, which itself is a highly effective optimization technique. In addition to that, Biotic Flower Pollination Algorithm is not complicated in structure and consists of only one non-iterational equation, which leads to its easy and fast implementation in every programming environment. In conclusion, the BFPA performance allows this technique to be used both for simple tasks as well as in more complicated optimization projects. To allow the replication of our result, the code of BFPA along with the short example of its usage was published on [16].

Since it has been observed that BFPA usually quickly finds the right region where the global minimum is located and most of the time looks for the most attractive point, further research concerning the BFPA's structure with special emphasis on exploitation process reinforcement, should be performed. It is also worth mentioning that obliteration of Levy flights in the BFPA excluded from the algorithm the best global solution found so far, which is quite often used in strengthening the exploitation process

and its inclusion to BFPA in alternative way might also be the theme of future studies.

**Acknowledgements** This work was partially financed (supported) by the Faculty of Physics and Applied Computer Science, AGH University of Science and Technology statutory tasks within subsidy of Ministry of Science and Higher Education.

### Compliance with ethical standards

**Conflict of interest** The authors declare that they have no conflict of interest.

**Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

### References

- Abdel-Baset M, Hezam I (2016) A hybrid flower pollination algorithm for engineering optimization problems. *Int J Comput Appl* 140:10–23
- Abdel-Basset M, Shawky LA (2018) Flower pollination algorithm: a comprehensive review. *Artif Intell Rev*. <https://doi.org/10.1007/s10462-018-9624-4>
- Abrol D (2012) Non bee pollinators–plant interaction. Springer, Dordrecht, pp 265–310. [https://doi.org/10.1007/978-94-007-1942-2\\_9](https://doi.org/10.1007/978-94-007-1942-2_9)
- Agarwal P, Mehta S (2018) Empirical analysis of five nature-inspired algorithms on real parameter optimization problems. *Artif Intell Rev* 50(3):383–439. <https://doi.org/10.1007/s10462-017-9547-5>
- Alyasseri Z, Khader A, Al-Betar M, Awadallah M, Yang XS (2018) Variants of the flower pollination algorithm: a review. Springer, Cham, pp 91–118. [https://doi.org/10.1007/978-3-319-67669-2\\_5](https://doi.org/10.1007/978-3-319-67669-2_5)
- Awad N, Ali M, Liang J, Qu B, Suganthan P (2016) Problem definitions and evaluation criteria for the cec 2017 special session and competition on single objective bound constrained real-parameter numerical optimization. Tech. rep., Nanyang Technological University, Singapore
- Cheng MY, Prayogo D (2014) Symbiotic organisms search: a new metaheuristic optimization algorithm. *Comput Struct* 139:98–112
- Chittka L, Thomson J, Waser N (1999) Flower constancy, insect psychology, and plant evolution. *Naturwissenschaften* 86(8):361–377
- Clerc M (2006) Particle swarm optimization. Translated from the French original., translated from the french original edn. London: ISTE. <https://doi.org/10.1002/9780470612163>
- Dahi Z, Mezioud C, Draa A (2016) On the efficiency of the binary flower pollination algorithm. *Appl Soft Comput* 47(C):395–414. <https://doi.org/10.1016/j.asoc.2016.05.051>
- Fausto F, Reyna-Orta A, Cuevas E, Andrade ÁG, Perez-Cisneros M (2019) From ants to whales: metaheuristics for all tastes. *Artif Intell Rev*. <https://doi.org/10.1007/s10462-018-09676-2>
- Gandomi AH, Alavi AH (2012) Krill herd: a new bio-inspired optimization algorithm. *Commun Nonlinear Sci Numer Simul* 17(12):4831–4845. <https://doi.org/10.1016/j.cnsns.2012.05.010>. <http://www.sciencedirect.com/science/article/pii/S1007570412002171>
- Kaur A, Pal S, Singh A (2017) New chaotic flower pollination algorithm for unconstrained non-linear optimization functions. *Int J Syst Assur Eng Manag*. <https://doi.org/10.1007/s13198-017-0664-y>
- Kent M (2013) Advanced biology. Advanced sciences, 2nd edn. Oxford University Press, Oxford
- Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. *Science* 220(4598):671–680. <https://doi.org/10.1126/science.220.4598.671>. <http://science.sciencemag.org/content/220/4598/671>
- Kopciwicz P, Lukasiak S (2018) FCA source code for fitting a function model to charge distributions. <http://home.agh.edu.pl/~slukasiak/pub/SourceFCA2018/>
- Lukasiak S, Kowalski P (2015) Study of flower pollination algorithm for continuous optimization. In: *Intelligent Systems'2014*. Springer, Cham, pp. 451–459
- Mirjalili S (2016) Sca: a sine cosine algorithm for solving optimization problems. *Knowl Based Syst* 96:120–133. <https://doi.org/10.1016/j.knsys.2015.12.022>. <http://www.sciencedirect.com/science/article/pii/S0950705115005043>
- Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. *Adv Eng Softw* 69:46–61. <https://doi.org/10.1016/j.advengsoft.2013.12.007>. <http://www.sciencedirect.com/science/article/pii/S0965997813001853>
- Nigdeli SM, Bekdaş G, Yang XS (2016) Application of the flower pollination algorithm in structural engineering. In: Yang XS, Bekdaş G, Nigdeli S (eds) *Metaheuristics and optimization in civil engineering. Modeling and optimization in science and technologies*, vol 7. Springer, Cham. [https://doi.org/10.1007/978-3-319-26245-1\\_2](https://doi.org/10.1007/978-3-319-26245-1_2)
- Pant S, Kumar A, Ram M (2017) Flower pollination algorithm development: a state of art review. *Int J Syst Assur Eng Manag* 8(2):1858–1866. <https://doi.org/10.1007/s13198-017-0623-7>
- Pavlyukevich I (2007) Levy flights, non-local search and simulated annealing. *J Comput Phys* 226(2):1830–1844. <https://doi.org/10.1016/j.jcp.2007.06.008>. <http://www.sciencedirect.com/science/article/pii/S002199910700263X>
- Rashedi E, Nezamabadi-pour H, Saryzadi S (2009) Gsa: a gravitational search algorithm. *Inf Sci* 179(13):2232–2248. <https://doi.org/10.1016/j.ins.2009.03.004>. <http://www.sciencedirect.com/science/article/pii/S0020025509001200> (special section on high order fuzzy sets)
- Salgotra R, Singh U (2017) Application of mutation operators to flower pollination algorithm. *Expert Syst Appl* 79:112–129. <https://doi.org/10.1016/j.eswa.2017.02.035>. <http://www.sciencedirect.com/science/article/pii/S0957417417301264>
- Saremi S, Mirjalili S, Lewis A (2017) Grasshopper optimisation algorithm: theory and application. *Adv Eng Softw* 105:30–47. <https://doi.org/10.1016/j.advengsoft.2017.01.004>. <http://www.sciencedirect.com/science/article/pii/S0965997816305646>
- Shambour M, Abusnaina A, Alsalibi A (2018) Modified global flower pollination algorithm and its application for optimization problems. *Interdiscip Sci Comput Life Sci*. <https://doi.org/10.1007/s12539-018-0295-2>
- Shi Y (2011) An optimization algorithm based on brainstorming process. *Int J Swarm Intell Res (IJSIR)* 2(4):35–62. <https://ideas.repec.org/a/igg/jsir00/v2y2011i4p35-62.html>
- Singh D, Singh U, Salgotra R (2018) An extended version of flower pollination algorithm. *Arab J Sci Eng*. <https://doi.org/10.1007/s13369-018-3166-6>

29. Sivanandam S, Deepa S (2007) Introduction to genetic algorithms, 1st edn. Springer, Berlin
30. Sorensen K (2015) Metaheuristics—the metaphor exposed. *Int Trans Oper Res* 22(1):3–18. <https://doi.org/10.1111/itor.12001>
31. Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *J Global Optim* 11(4):341–359. <https://doi.org/10.1023/A:1008202821328>
32. Tan Y, Zhu Y (2010) Fireworks algorithm for optimization. In: Tan Y, Shi Y, Tan KC (eds) *Advances in swarm intelligence*. Springer, Berlin, pp 355–364
33. Wagih KA (2015) An improved flower pollination algorithm for solving integer programming problems. *Appl Math Inf Sci Lett* 3:31–37
34. Wang R, Zhou Y (2014) Flower pollination algorithm with dimension by dimension improvement. *Math Probl Eng* 2014:481791. <https://doi.org/10.1155/2014/481791>
35. Welch B (1947) The generalization of ‘student’s’ problem when several different population variances are involved. *Biometrika* 34(1/2):28–35. <http://www.jstor.org/stable/2332510>
36. Xu S, Wang Y, Huang F (2017) Optimization of multi-pass turning parameters through an improved flower pollination algorithm. *Int J Adv Manuf Technol* 89(1):503–514. <https://doi.org/10.1007/s00170-016-9112-4>
37. Yang XS (2012) Flower pollination algorithm for global optimization. In: Durand-Lose J, Jonoska N (eds) *Unconventional computation and natural computation*. Springer, Berlin, pp 240–249

**Publisher’s Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations