International Journal of Computational Intelligence and Applications
Vol. 17, No. 1 (2018) 1850002 (27 pages)
© World Scientific Publishing Europe Ltd.
DOI: 10.1142/S1469026818500025

Focus Group: An Optimization Algorithm Inspired by Human Behavior

Edris Fattahi[*,§], Mahdi Bidar[†,¶] and Hamidreza Rashidy Kanan[‡,||]

[*]*Department of Computer Engineering, Marivan Branch*
Islamic Azad University, Marivan, Iran

[†]*Department of Computer Science*
University of Regina, Regina, SK, Canada S4S 0A2

[‡]*Department of Computer Engineering*
Shahid Rajaee Teacher Training University, Tehran, Iran
[§]*edris.fattahi@gmail.com*
[¶]*bidarzom@uregina.ca*
[||]*h.rashidykanan@srttu.edu*

Received 3 February 2017
Revised 31 October 2017
Accepted 16 January 2018
Published 12 March 2018

This paper presents a novel optimization algorithm, namely focus group (FG) algorithm, for solving optimization problems. The proposed algorithm is inspired by the behavior of group members to share their ideas (solutions) about a specific subject and trying to improve the solutions based on the cooperation and discussion. In the proposed algorithm, all the members present their solutions about the subject and all the suggested solutions proportional to their fitness, leave impact on the other solutions and incline them towards themselves. While trying to improve the quality of the candidate solutions, they converge to the optimal solution. To improve the performance of the proposed algorithm, two genetic operators are incorporated into the algorithm. The proposed algorithm is evaluated using several constrained and unconstrained benchmark functions commonly used in the area of optimization. Experimental results, in comparison with different well-known evolutionary techniques, confirm the high performance of the proposed algorithm in dealing with the optimization problems.

Keywords: Optimization; meta-heuristic algorithms; search algorithms; focus group (FG) optimization algorithm.

1. Introduction

In recent decades, scientists in almost every area have been grappling with high-dimensional and complex problems. Since the problem space increases exponentially with the problem size, the mathematical and classical methods cannot provide a suitable solution efficiently.[1] Therefore, the practical approach is utilizing the

[||]Corresponding author.

stochastic methods which are divided into heuristic and meta-heuristic algorithms. The idea of using meta-heuristic methods is to find the qualified solution in a reasonably practical time. However, there is no guarantee that the best solutions can be found. Two main components of any meta-heuristic algorithms are diversification and intensification. Diversification means to generate diverse solutions to globally explore the search space, while intensification means to focus on the search in a local region by exploiting the information that a current good solution is found in this region.[1,2]

The meta-heuristic algorithms do not have limitations in using source of inspiration. Different meta-heuristic optimization algorithms have been proposed during the recent decades. *Simulated annealing* (SA) which is inspired by annealing process of substances such as metals[3] is one of the earliest metaphor-based meta-heuristics (although the term "meta-heuristic" did not exist at that time[4]). Classification of the meta-heuristic algorithms can be carried out in different ways. Considering the inspiration source, meta-heuristic algorithms can be classified into two categories:

- evolutionary algorithms (EAs);
- swarm-based algorithms.

EAs which are inspired by the biological evolutions, by using an iterative process try to make progress in dealing with the optimization problems. The most famous paradigm of EAs is *genetic algorithm* (GA) which was developed based on Darwin's theory of evolution and survival of fittest in biological systems.[5,6] Two main operators of GA are crossover and mutation. Utilizing these operators in an iterative process, GA improves the quality of the candidate solutions in order to solve an optimization problem. Differential evolution (DE)[7] and evolutionary strategy (ES)[8] are other well-known paradigms of the evolutionary algorithms which use biological mechanisms such as crossover and mutation. Harmony search (HS) proposed by Geem *et al.,*[9] is another evolutionary algorithm which is inspired by musicians' behavior for searching the better harmonies.

Most of the proposed meta-heuristic optimization algorithms belong to the swarm-intelligence-based algorithms. Swarm-intelligence refers to the collective behavior of population of simple self-organized agents such as ant, bee, etc. in nature which are interacting locally with each other. This collective behavior and local interaction results in global behavior and eventually swarm-intelligence emerging in a group of simple agents.[1,2] Particle swarm optimization (PSO)[10] is one of the most famous algorithms in this area. PSO which was first proposed by Eberhart and Kennedy is inspired by social behavior of bird flocking or fish schooling. Their collective behavior for sharing their current best solution and updating the global best solution results in finding the optimal solution. Ant colony optimization (ACO)[11] is another well-known swarm-intelligence optimization algorithm. This algorithm is inspired by the collective foraging behavior of ants in finding the shortest path between the discovered food source and their nest. Leaving a chemical substance namely pheromone (which evaporates over time) and choosing the path with more of pheromone on the ground is the main characteristic and main governing rule of every ant and ant colony in nature. Artificial bee colony optimization (ABC) algorithm is another famous swarm-intelligence algorithm which is inspired by the behavior of honey bees in discovering the area for finding new and rich food sources in order to maximize the nectar stored in their hive.[12] Firefly algorithm (FA)[13] is another swarm optimization algorithm which is inspired by the behavior of the fireflies in emitting light in order to attract other fireflies. Recently some other meta-heuristic optimization algorithms have been proposed by the researchers. Some of them are cuckoo optimization algorithm (COA) which is inspired by laying eggs and breeding characteristics of cuckoos,[14] gravitational search algorithm (GSA) which is proposed based on the law of gravity and mass interaction,[15] intelligent water drops algorithm which is inspired by the flow of water into the sea,[16] brain storm optimization (BSO) which is proposed based on human brainstorming process,[17] krill herd (KH) optimization algorithm which is inspired by herding behavior of krill individuals,[18] dolphin echolocation algorithm (DEA) which is proposed based on navigation ability of dolphins,[19] Egyptian vulture optimization (EVO) algorithm which is inspired by Egyptian vulture skills in acquiring foods,[20] gray wolf optimizer (GWO) which mimics the leadership hierarchy and hunting mechanism of gray wolves in nature,[21] ant lion optimizer (ALO) which mimics the hunting mechanism of antlions in nature,[22] water wave optimization (WWO) which is inspired by the shallow water wave theory[23] and optics-inspired optimization (OIO) which is based on the optical characteristics of concave and convex mirrors.[24] Moreover, other new meta-heuristic optimization algorithms, such as interior search algorithm (ISA) which is inspired by interior design and decoration,[25] heat transfer search (HTS) which is inspired by the law of thermodynamics and heat transfer,[26] dragonfly algorithm[27] which is inspired by static and dynamic swarming behaviors of dragonflies in nature and multi-verse optimizer[28] whose main inspirations are based on three concepts in cosmology: white hole, black hole and wormhole, have been recently introduced.

It is shown by many researchers that meta-heuristic optimization algorithms have been widely employed for solving complex optimization problems such as optimization of objective functions,[29,30] power systems operations and control,[31–33] chemical processes,[34] job scheduling problems,[35] pattern recognition,[36–41] image processing,[42–51] etc.

Although the aforementioned meta-heuristic algorithms provide satisfactory results, they are not supposed to deal with all optimization tasks effectively.[52] Hence, introducing new meta-heuristic optimization algorithms are welcomed. The particular objectives of developing the new meta-heuristic algorithms are to address problems faster, to solve large problems and to obtain more robust techniques.[53]

This paper introduces a novel evolutionary technique called *focus group* (FG) algorithm which is inspired by the improvement and evolution process of the ideas (solutions) of group members in cooperating with each other. Some meta-heuristic algorithms like GA or DE utilize randomized operators to select the parents (i.e., the solutions that will influence the evolution of the population). However, the proposed FG optimization algorithm like most of the other well-known meta-heuristic optimization algorithms which rely on a population of solutions that are influenced by other normally (but not necessarily always) better solutions to evolve, relies on all solutions that each of them affects or is affected by the other solutions. In other words, in the proposed FG optimization algorithm, by discussion on a specific subject, each participant affects or is influenced by the solutions presented by other participants in an iterative procedure until the best solution for the problem is obtained. This process has been our inspiration source for introducing a new evolutionary optimization algorithm. In the proposed optimization algorithm, the evolutionary procedure of improving solutions of the group participants has been modeled as an optimization strategy. To improve the performance of the proposed algorithm, two genetic operators are incorporated into the algorithm. Feasibility and effectiveness investigation for the proposed FG optimization algorithm is conducted using several constrained and unconstrained standard benchmark functions commonly used in the area of optimization. The system performance is compared with the performances of some well-known meta-heuristic optimization algorithms. Our obtained results confirm the high performance of the proposed method in dealing with the optimization problems.

In the following, Sec. 2 briefly presents the FG methodology and constraints. The proposed FG algorithm is introduced in Sec. 3. In Sec. 4, the experimental results are demonstrated. A comparative study is presented in Sec. 5. Finally, Sec. 6 concludes the paper.

2. Focus Group

FG is a data collection procedure in the form of a carefully planned group discussion among almost 10 people plus a moderator (facilitator) and an observer (note taker), in order to obtain diverse solutions and perceptions on a topic of interest in a relaxed, permissive environment that fosters the expression of different points of view, with no pressure for consensus.[54] According to the above definition, FG is a group consisting of members who discuss about a certain subject or problem and share their solutions in order to obtain the best solution about the mentioned subject. The main characteristic of FG is its ability to produce and improve the solutions based on cooperation of the members. In FG, solutions of all members will be taken into consideration by all members and a better solution affects the worse ones to improve them. By repeating this procedure, the best solution will be obtained. Two main governing rules which must be considered in FG are as follows:

- Each time, only one member is allowed to talk. Different ideas (solutions) are all valuable but with different values. The better solutions possess higher values and have greater impact on the solutions of the other members.

- According to the mentioned general definition of FG, in addition to those members with different solutions, there are also two other members called facilitator and note taker. The facilitator is the supervisor of the group and the note taker is responsible for recording all the events happening during the meeting.

The main responsibilities of the facilitator and note taker are as follows:

- Each group has a main goal which is defined on the particular subject and the solutions of the members should be in the defined framework of the problem. One of the main responsibilities of facilitator is to monitor these solutions to be in the defined framework.
- Members with better solutions are likely to dominate the meeting and cause decrease in the participation rate of other members. The responsibility of the note taker is to record such events in each iteration. The facilitator takes these records into consideration and in order to utilize all the solutions to reach the best solution, gives a higher value to the less important solutions and makes them to participate more actively in the discussion. In FG, all the members should participate in the discussion and it is not common that discussion is handled by only the participants with more important solutions.

3. The Proposed FG Optimization Algorithm

In this section, we introduce our novel FG optimization algorithm. The proposed FG optimization algorithm is designed based on obtaining the steps for a best solution in a focus group. In the proposed algorithm, all the members present their solutions on the subject of the meeting. Better solutions possess higher values, so incline the weaker solutions towards themselves and by affecting the weaker solutions try to improve and enhance them. The weak solutions also affect the other solutions; however these effects are less than those of the better solutions. This trend will be continued until they obtain the best solution. During the discussion, while the solutions gradually converge to the best solution, note taker records all the events and based on the records, the facilitator supervises the meeting in the framework of the problem. The facilitator also prevents the meeting from early convergence to optimal solution which strengthens the exploration property of the algorithm.

The pseudo-code of the proposed FG optimization algorithm is shown in Fig. 1.

In the initial step, all participants of a focus group present their solutions considering the problem framework and their fitness values are calculated based on the defined fitness function. In this step, the best solution for each participant is their first solution and the note taker records the first best solution. The global best solution in kth iteration is NBC^k. In each iteration, the impact of the best solutions of the other participants on the next solution of the current participant is calculated

```
Initialization
while any termination criterion is not met
        for $i = 1$ to $N$
            IC = getCindex(PBC)
            Calculate impact of each participant's solution:
            $\mathrm{PIm}_i^{k+1} = w \times \mathrm{PIm}_i^k + \sum_{j=1}^{N} (\mathrm{IC}_j \times \mathrm{Rnd} \times (\mathrm{PBI}_j^k - \mathrm{PI}_i^k))$
            Apply impact limits on $\mathrm{PIm}_i^{k+1}$
            Update $\mathrm{PI}_i^{k+1}$ based on $\mathrm{PIm}_i^{k+1}$
            Apply limits on $\mathrm{PI}_i^{k+1}$ by facilitator based on the predefined framework
            Apply crossover operator
            Evaluate the $\mathrm{PC}_i^{k+1}$
            Update $\mathrm{PBI}_i^{k+1}$ based on $\mathrm{PC}_i^{k+1}$ and $\mathrm{PBI}_i^k$
        end for
    Update $\mathrm{NBC}^k$
    $k = k + 1$;
    Apply replacement operator
end while
```

Notes: PIm_i^k : Impact of ith participant in kth iteration; PBI_i^k : Best solution of ith participant in kth iteration; PI_i^k : Solution of ith participant in kth iteration; NBC^k: Best cost in kth iteration; IC_i: Impact coefficient (IC) belonging to ith participant; w: Inertia weight; PC_i^k : Cost of ith participant in kth iteration; PBC: Best cost of all participants; Rnd: Random number; getCindex: A function that returns impact coefficient based on PBC.

Fig. 1. The pseudo-code of the proposed FG optimization algorithm.

based on Eq. (1):

$$\mathrm{PIm}_i^{k+1} = w \times \mathrm{PIm}_i^k + \sum_{j=1}^{N} (\mathrm{IC}_j \times \mathrm{Rnd} \times (\mathrm{PBI}_j^k - \mathrm{PI}_i^k)), \tag{1}$$

where PIm_i^k is the impact of the other solutions on the participant i in kth iteration, PI_i^k is the solution of the participant i in kth iteration, PBI_j^k is the best solution of the participant j so far until iteration k and IC_j is the impact coefficient of the participant j which should be determined based on the best obtained cost for the participant j. Also, w is the inertia weight which is a numerical value in the interval [0, 1]. It should be mentioned that the best solution of ith participant in the first iteration is its first solution. In the following iterations, if it gets better solution with better fitness value, the previous solution will be replaced by the new better solution and this is the main trend of the algorithm from the beginning to the end.

Figure 2 shows the vector illustration of Eq. (1) in two-dimensional spaces.

The constraints are applied to PIm_i^k [according to Eq. (2)] so that the max() and min() functions determine the maximum and the minimum bounds of

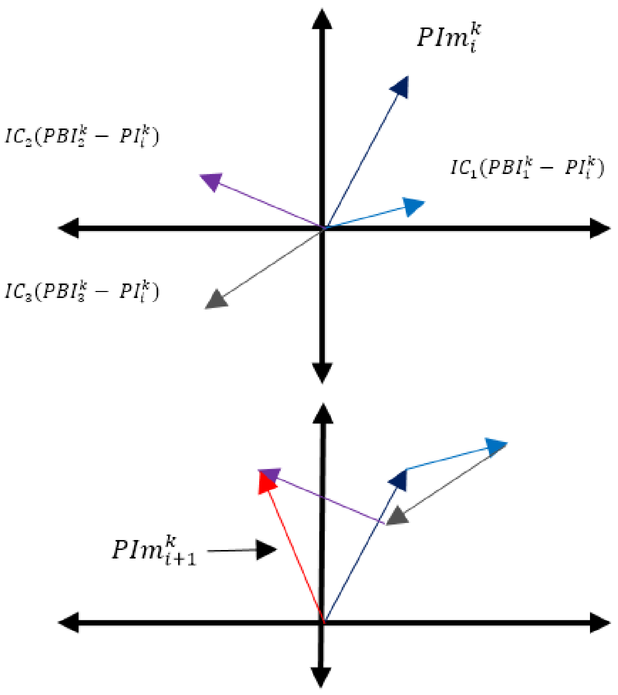

Fig. 2. The vector illustration of Eq. (1) in two-dimensional spaces.

their inputs:

$$\mathrm{PIm}_i^k = \begin{cases} \max(\mathrm{PIm}_i^k, \text{lower bound of the impact}), \\ \min(\mathrm{PIm}_i^k, \text{upper bound of the impact}). \end{cases} \tag{2}$$

Determination of lower and upper bounds of the impact is based on the lower and the upper bounds of benchmark functions or real-world problems, respectively. The solution of the participant i is updated based on the calculated impact of the other solutions (previous step) using Eq. (3):

$$\mathrm{PI}_i^{k+1} = \mathrm{PI}_i^k + \mathrm{PIm}_i^{k+1}. \tag{3}$$

The constraints over PI_i^k are calculated by the facilitator according to Eq. (4). Here, the max() and min() functions determine the upper and the lower bounds of their inputs:

$$\mathrm{PI}_i^k = \begin{cases} \max(\mathrm{PI}_i^k, \text{the lower bound of the problem}), \\ \min(\mathrm{PI}_i^k, \text{the upper bound of the problem}). \end{cases} \tag{4}$$

In the updating process of PBI_i^{k+1} in $(k+1)$th iteration, if the fitness of the ith participant is better than its best found fitness in previous iterations, the previous solution will be replaced by the new one.

Figure 3 indicates the vector illustration of Eq. (3) in two-dimensional spaces.

After this step, the cost of each solution is evaluated based on the fitness function and considering the new solution of the participants, the global best solution is updated. If the termination criterion is met, the algorithm is completed, otherwise, it returns.

In order to improve the performance of the proposed FG optimization algorithm, we incorporate crossover operator after updating participants and replacement operator after the end of each iteration separately. Crossover operator is a genetic operator that combines two chromosomes in order to produce better chromosomes. Based on a vectorized version of the crossover operator which is used in DE algorithm, an adaptive vectorized crossover operator is employed in our study. The crossover is controlled by a crossover probability "cr," and based on this operator, the mth parameter of the participant i is replaced by a random value in [0, 1]. Equation (5) shows this operation:

$$\text{participant}_{i,m} = \begin{cases} \text{participant}_{r,m} & \text{rand}_{i,m} < \text{cr}, \quad \text{cr} = 0.2 * \text{BestCost}, \\ \text{participant}_{i,m} & \text{otherwise}. \end{cases} \tag{5}$$

where $r \in \{1, 2, 3, \ldots, n\} - \{i\}$. Decreasing the best cost and therefore decreasing "cr" will cause more participants to remain unchanged in the next generation.

For replacement operation, increasing the replacement rate will cause early convergence and also increase the risk of being trapped in local optimums. In contrast, decreasing the replacement rate will decrease the convergence speed and make it perform like the standard FG algorithm. This parameter was set empirically as 0.05 to provide the balance between the mentioned characteristics. Therefore, after each iteration of the proposed algorithm, 5% of the worst participants are eliminated and replaced by the best participants. It should be mentioned that when a new solution is produced by replacement operator, the "PIm" values of all participants are recalculated by getCindex function in the next iteration.

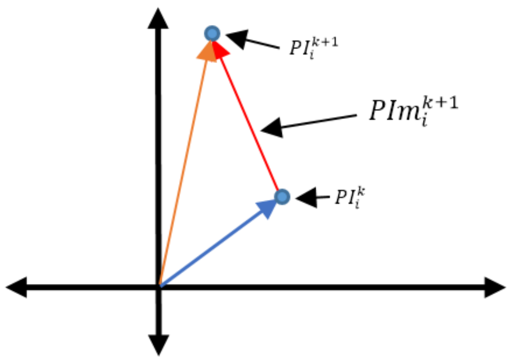

Fig. 3. The vector illustration of Eq. (3) in two-dimensional spaces.

4. Experimental Results

In order to evaluate the performance of the proposed FG optimization algorithm, an extensive experimental investigation is conducted using several well-known and standard benchmark functions. The utilized benchmark functions contain both constrained and unconstrained problems. As a clarifier example, we use the peak function to visually present the procedure of the proposed FG optimization algorithm in Sec. 4.1. The performances of the proposed FG optimization algorithm are compared with different famous meta-heuristic algorithms.

4.1. *The clarifier example*

In this subsection, in order to pre-evaluate the performance of the proposed FG optimization algorithm and to illustrate the running steps of finding the optimal solution, we utilize the peak function [Eq. (6)]. The problem space is $[-2, 2] \times [-2, 2]$ (Ref. 2) and the goal is to find the parameter values that result in a minimum output:

$$f(x) = x_1 \times e^{-(x_1^2 + x_2^2)}. \tag{6}$$

The running steps of the proposed FG optimization algorithm are displayed in Figs. 4(a)–4(f). The solutions of the participants are indicated by the blue points. In the initialization step, 20 solutions are distributed randomly in the defined problem space.

Figure 4(a) shows the initialization of the proposed FG optimization algorithm. In this step, the solutions are randomly distributed in the problem space. Then, they start to affect each other based on their fitness values. As Figs. 4(b)–4(f) show, the positions of the solutions' concentration go towards the global minimum with increasing the iterations.

4.2. *Standard benchmark functions*

In this experiment, several standard well-known constrained and unconstrained benchmark functions are employed to evaluate the performance of the proposed FG optimization algorithm.

4.2.1. *Unconstrained benchmark functions*

In this sub-subsection, 20 unconstrained benchmark functions which are utilized in the optimization literature are presented. These unconstrained benchmark functions which are indicated in Table 1, have different characteristics like many local minima, valley-shaped, bowl-shaped, steep drops and plate-shaped. The number of design variables and the corresponding ranges are different for each problem.

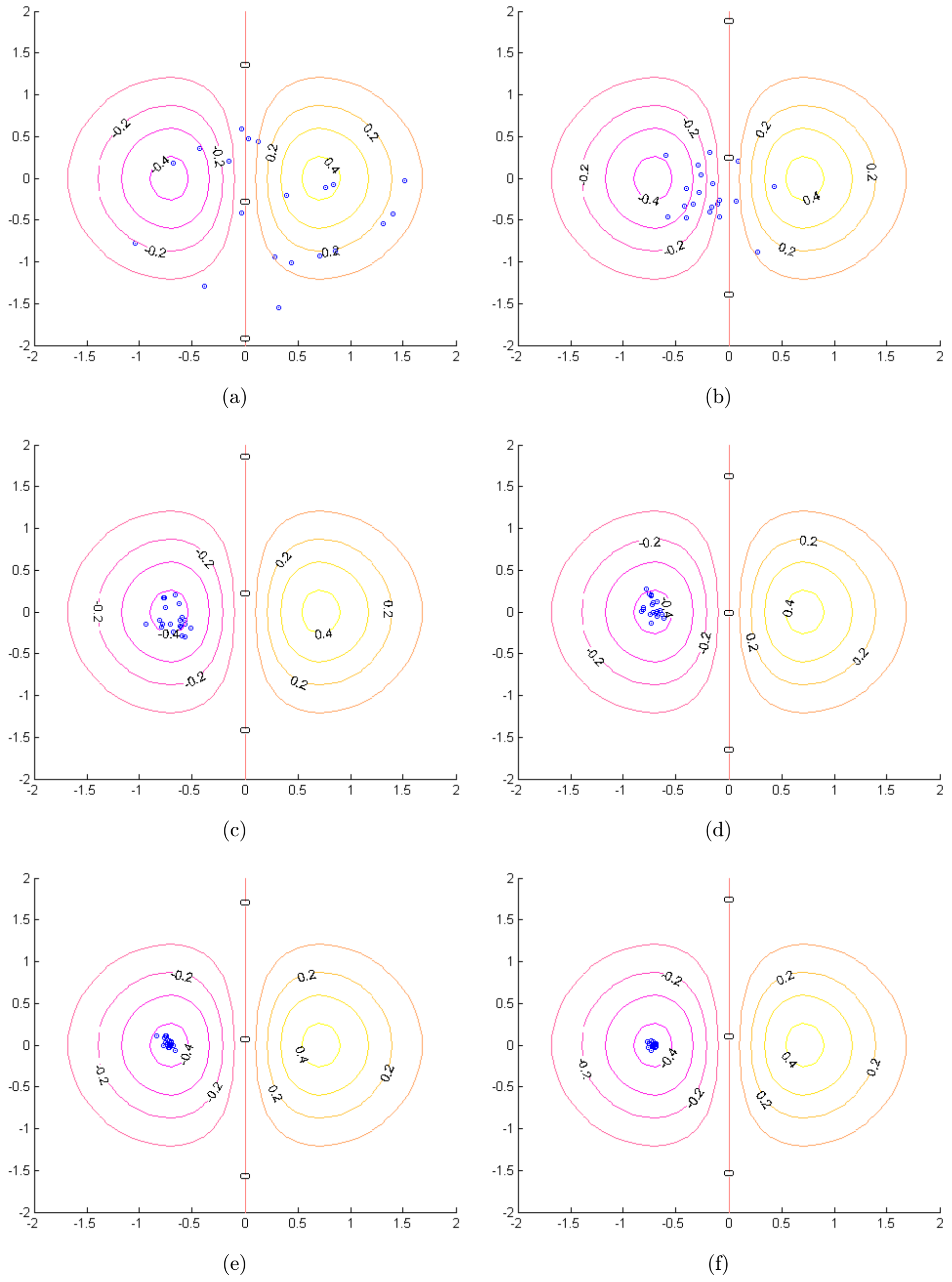

Fig. 4. Running steps of the proposed FG optimization algorithm to find the global best (minimum) of the peak function: (a) after one iteration, (b) after five iterations, (c) after 10 iterations, (d) after 15 iterations, (e) after 20 iterations and (f) after 25 iterations.

Table 1. The 20 unconstrained benchmark functions used in our experiments.

ID	Name	Dim (n)	Function	Domain	Global Minimum	Type
F1	Ackley	20	$f(x) = -20\exp\left(-0.2\sqrt{n^{-1}\sum_{i=1}^{n} x_i^2}\right) -\exp\left(n^{-1}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	$x_i \in [-32.768, 32.768]$	0	Many local minima
F2	Griewank	20	$f(x) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \Pi_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)$	$x_i \in [-600, 600]$	0	Many local minima
F3	Levy	20	$f(x) = \sin^2(\pi\omega_1) + \sum_{i=1}^{n-1}(\omega_i - 1)^2[1 + 10\sin^2(\pi\omega_i + 1)] + (\omega_n - 1)^2[1 + \sin^2(2\pi\omega_n)]$, where $\omega_i = 1 + \dfrac{x_i - 1}{4}$, for all $i = 1, \ldots, n$	$x_i \in [-10, 10]$	0	Many local minima
F4	Rastrigin	20	$f(x) = 10n + \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i))$	$x_i \in [-5.12, 5.12]$	0	Many local minima
F5	Schwefel	20	$f(x) = 418.9829n - \sum_{i=1}^{n} x_i \sin(\sqrt{\lvert x_i\rvert})$	$x_i \in [-500, 500]$	0	Many local minima
F6	Dixon–Price	20	$f(x) = (x_1 - 1)^2 + \sum_{i=2}^{n} i(2x_i^2 - x_{i-1})^2$	$x_i \in [-10, 10]$	0	Valley-shaped
F7	Rosenbrock	20	$f(x) = \sum_{i=1}^{n-1}(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	$x_i \in [-5, 10]$	0	Valley-shaped
F8	Sphere	20	$f(x) = \sum_{i=1}^{n} x_i^2$	$x_i \in [-5.12, 5.12]$	0	Bowl-shaped
F9	Powell	20	$f(x) = \sum_{i=1}^{n/4}[(x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 +(x_{4i-3} + 2x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4]$	$x_i \in [-4, 5]$	0	Other
F10	Michalewicz	10	$f(x) = -\sum_{i=1}^{n}\sin(x_i)\sin^{20}\left(\frac{ix_i^2}{\pi}\right)$	$x_i \in [0, \pi]$	−9.66	Steep drops
F11	Colville	4	$f(x) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2 + 10.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1)$	$x_i \in [-10, 10]$	0	Other
F12	Shubert	2	$f(x) = \left(\sum_{i=1}^{5} i\cos((i+1)x_1 + i)\right)\left(\sum_{i=1}^{5} i\cos((i+1)x_2 + i)\right)$	$x_i \in [-10, 10]$	−186.73	Many local minima
F13	Six-hump camel	2	$f(x) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$	$x_1 \in [-3, 3]$, $x_2 \in [-2, 2]$	−1.03	Valley-shaped

Table 1. (*Continued*)

ID	Name	Dim (n)	Function	Domain	Global Minimum	Type
F14	Bohachevsky 1	2	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$	$x_i \in [-100, 100]$	0	Bowl-shaped
F15	De Jong N.5	2	$f(x) = \left(0.002 + \sum_{i=1}^{25} \frac{1}{i + (x_1 - a_{1i})^6 + (x_2 - a_{2i})^6} \right)^{-1}$, where $a = \begin{pmatrix} -32 & -16 & 0 & \dots & 9 & 16 & 32 \\ -32 & -32 & -32 & \dots & 32 & 32 & 32 \end{pmatrix}$	$x_i \in [-65.536, 65.536]$	0.99	Steep drops
F16	Easom	2	$f(x) = -\cos(x_1)\cos(x_2)\exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$	$x_i \in [-100, 100]$	-1	Steep drops
F17	Matyas	2	$f(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$	$x_i \in [-10, 10]$	0	Plate-shaped
F18	Beale	2	$f(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$	$x_i \in [-4.5, 4.5]$	0	Other
F19	Goldstein–Price	2	$f(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	$x_i \in [-2, 2]$	3	Other
F20	Forrester–Sobester–Keane (2008)	1	$f(x) = (6x - 2)^2 \sin(12x - 4)$	$x \in [0, 1]$	-6.02	Other

4.2.2. *Constrained benchmark functions*

In this sub-subsection, we present 24 constrained benchmark functions which are utilized in the optimization literature. The characteristics of these constrained benchmark functions which are shown in Table 2 are different from the viewpoints of being linear, nonlinear, quadratic, cubic and polynomial. The number of design variables and the corresponding ranges are different for each problem.[55]

4.3. *Determination of parameters*

Similar to the other meta-heuristic optimization algorithms, the proposed FG optimization algorithm has parameters which should be accurately tuned to improve the performance of the algorithm. Due to the importance of the accurate tuning of these parameters, an extensive experiment is conducted to set these parameters with appropriate values.

Table 2. The 24 constrained benchmark functions used in our experiments and their characteristics.

Function	N	ρ (%)	LI	NI	LE	NE	ac	O	Type
G1	13	0.0111	9	0	0	0	6	−15	Quadratic
G2	20	99.9971	0	2	0	0	1	−0.8036	Nonlinear
G3	10	0.0000	0	0	0	1	1	−1.0005	Polynomial
G4	5	52.1230	0	6	0	0	2	−30665.5	Quadratic
G5	4	0.0000	2	0	0	3	3	5126.496	Cubic
G6	2	0.0066	0	2	0	0	2	−6961.81	Cubic
G7	10	0.0003	3	5	0	0	6	24.3062	Quadratic
G8	2	0.8560	0	2	0	0	0	−0.09582	Nonlinear
G9	7	0.5121	0	4	0	0	2	680.63	Polynomial
G10	8	0.0010	3	3	0	0	6	7049.248	Linear
G11	2	0.0000	0	0	0	1	1	0.7499	Quadratic
G12	3	4.7713	0	1	0	0	0	−1	Quadratic
G13	5	0.0000	0	0	0	3	3	0.0539	Nonlinear
G14	10	0.0000	0	0	3	0	3	−47.7649	Nonlinear
G15	3	0.0000	0	0	1	1	2	961.715	Quadratic
G16	5	0.0204	4	34	0	0	4	−1.9051	Nonlinear
G17	6	0.0000	0	0	0	4	4	8853.539	Nonlinear
G18	9	0.0000	0	13	0	0	6	−0.86602	Quadratic
G19	15	33.4761	0	5	0	0	0	32.6556	Nonlinear
G20	24	0.0000	0	6	2	12	16	0.20498	Linear
G21	7	0.0000	0	1	0	5	6	193.7245	Linear
G22	22	0.0000	0	1	8	11	19	236.4309	Linear
G23	9	0.0000	0	2	3	1	6	−400.055	Linear
G24	2	79.6556	0	2	0	0	2	−5.50801	Linear

Notes: N is the number of decision variables, ρ is the estimated ratio between the feasible region and the search space, LI is the number of linear inequality constraints, NI is the number of nonlinear inequality constraints, LE is the number of linear equality constraints, NE is the number of nonlinear equality constraints, ac is the number of active constraints at the optimum solution and O is the global optimum result.

Due to the random nature and stochastic behavior of the proposed FG optimization algorithm and also other meta-heuristic optimization algorithms, their real performance cannot be judged based on the result of a single run. In other words, several experiments with different initialization conditions should be carried out to obtain the real performance of an algorithm. So, in this research the results are obtained in 50 trials. Also in all the experiments, the number of iterations and the number of participants are set as 200 and 20, respectively.

4.3.1. *Impact coefficients*

As stated before, in the proposed FG optimization algorithm all the solutions are important and all of them have impact on the other solutions. However, better solutions are more important and therefore should have greater impact on the other solutions. In this research, the exponential distributed random numbers are allocated for impact coefficients. In other words, by generating a vector of random values with the length of the number of participants and considering the fitness of the participants, the highest value is associated to the best solution, the second highest value to the second best solution and so on. For instance, when our purpose is to minimize a problem, those solutions with less cost (calculated by the cost function) should be given a bigger IC value to increase their impact on the other solutions. Assume the number of participants is 10, if the mentioned distribution generates 10 numbers with the average of 0.5, allocation of these values to the participants is tabulated in Table 3.

4.3.2. *Tuning the inertia weight*

Another parameter that affects the performance of the proposed FG optimization algorithm is inertia weight (w). The best value for w is obtained from the extensive experiments on the mentioned unconstrained benchmark functions. In order to facilitate the investigation, the obtained results are normalized using Eq. (7):

$$X_{i,\text{normalized}} = 1 - \frac{(X_i - X_{\min})}{(X_{\max} - X_{\min})}, \tag{7}$$

where $X_{i,\text{normalized}}$ is the normalized value of solution i, X_i is the fitness value of solution $i, X_{\min}$ and $X_{\max}$ are the minimum and the maximum fitness values of the

Table 3. Assigning the generated random numbers to the participants considering their fitness values.

	PI_1 Solution	PI_2 Solution	PI_3 Solution	PI_4 Solution	PI_5 Solution	PI_6 Solution	PI_7 Solution	PI_8 Solution	PI_9 Solution	PI_{10} Solution
Fitness value	48	19	30	22	5	16	45	11	41	36
IC value	0.017	0.551	0.124	0.267	1.695	0.848	0.019	1.024	0.034	0.120

Table 4. Normalized results of the proposed FG optimization algorithm using different values of w.

	w									
ID	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
F1	0	0.038	0.144	0.227	0.329	0.560	0.805	0.909	1	0.299
F2	0	0.067	0.462	0.543	0.705	0.818	0.917	1	0.937	0.639
F3	0	0.145	0.514	0.721	0.805	0.917	0.944	0.957	1	0.882
F4	0	0.119	0.187	0.468	0.623	0.837	0.961	1	0.705	0.197
F5	0.005	0.569	0	0.396	0.441	0.735	1	0.896	0.241	0.295
F6	0	0.640	0.650	0.780	0.937	0.987	0.998	1	0.999	0.934
F7	0	0.328	0.273	0.532	0.727	0.923	0.978	0.999	1	0.992
F8	0.064	0	0.236	0.448	0.754	0.932	0.990	1	0.999	0.656
F9	0	0.122	0.253	0.613	0.861	0.957	0.982	0.999	1	0.898
F10	0	0.076	0.098	0.293	0.468	0.560	0.763	0.933	1	0.337
F11	0	0.158	0.155	0.555	0.846	0.931	0.920	0.998	1	0.937
F12	0	0.072	0.559	0.407	0.482	0.806	0.985	1	0.998	0.959
F13	0	0.064	0.951	0.969	1	1	1	1	0.999	0.984
F14	0	0.94046	0.992726	1	1	1	1	1	0.999	0.882
F15	0.317	0.514	0.455	0	0.142	0.172	0.250	0.435	0.226	1
F16	0	0.186	0.517	0.710	1	1	1	1	0.999	0.857
F17	0.111	0	0.816	0.659	1	1	1	1	0.999	0.973
F18	0.262	0	0.256	0.546	0.761	0.999	0.999	0.876	1	0.751
F19	0.019	0.494	0	0.502	0.676	1	1	1	1	0.999
F20	0	0.878	0.701	0.419	0.890	0.983	0.928	0.885	1	0.873
AVG	0.0389	0.270523	0.410986	0.5394	0.72235	0.85585	0.921	0.94435	0.90505	0.7672

found solutions, respectively. The normalized results of this experiment are shown in Table 4. These values indicate the influences of the different values of w on the performance of the proposed FG optimization algorithm.

It can be observed from Table 4 that the best performance of the proposed FG optimization algorithm is achieved when w is set as 0.7, 0.8 and 0.9. Therefore, in the rest of our experiments the value of w is set as 0.8.

4.4. ***Experiments***

In this subsection, we assess the performance of the proposed FG optimization algorithm with replacement operator using the mentioned standard benchmark functions.

4.4.1. *Experiments on unconstrained benchmark functions*

Table 5 shows the obtained results of the proposed FG algorithm for the unconstrained benchmark functions in 50 trials with 200 iterations.

It can be observed from Table 5 that the proposed FG optimization algorithm performs well in finding the optimal solutions which confirm the high performance of the proposed FG optimization algorithm.

Table 5. The obtained results of the proposed FG algorithm for the unconstrained benchmark functions in 50 trials with 200 iterations.

Function ID	Algorithm	Best	Mean	Median	Worst	Std. Dev
F1	FG	0.005	0.019	0.011	0.021	0.070
F2	FG	0.003	0.137	0.056	0.427	0.158
F3	FG	0.0004	0.364	0.272	1.728	0.527
F4	FG	3.986	6.810	6.507	9.961	1.817
F5	FG	5494.509	6150.078	6178.371	6746.840	390.683
F6	FG	0.680	0.757	0.724	0.975	0.089
F7	FG	11.787	25.002	20.361	78.927	19.591
F8	FG	6.16E−05	0.5E−03	0.3E−03	0.002	0.6E−03
F9	FG	0.021	0.053	0.050	0.103	0.027
F10	FG	−9.615	−9.264	−9.459	−8.621	0.410
F11	FG	0.036	0.597	0.291	2.826	0.841
F12	FG	−186.730	−186.688	−186.730	−186.527	0.067
F13	FG	−1.031	−1.031	−1.031	−1.031	7.51E−14
F14	FG	4.93E−16	1.70E−15	3.84E−15	1.20E−11	4.11E−12
F15	FG	1.992	8.075	7.873	12.670	2.869
F16	FG	−1	−1	−1	−1	1.04E−11
F17	FG	3.13E−27	3.94E−25	1.79E−26	2.42E−24	7.90E−25
F18	FG	1.06E−15	1.71E−12	8.48E−14	7.65E−12	2.92E−12
F19	FG	3	3	3	3	4.63E−14
F20	FG	−6.020	−6.020	−6.020	−6.020	4.14E−11

To evaluate the capability of the proposed FG optimization algorithm on unconstrained standard benchmark functions, its performance is compared with six well-known meta-heuristic optimization algorithms containing GA,[56] PSO,[57] DE,[7] ABC,[12] teaching–learning-based optimization (TLBO)[58,59] and KH.[18] The controlling parameters of the compared algorithms in this experiment are tabulated in Table 6.

The obtained results of the proposed FG optimization algorithm together with the results of the mentioned meta-heuristic optimization algorithms are tabulated in Table 7. It should be mentioned that these obtained results are the average of the normalized results in 50 trials.

In order to make a better comparison of the proposed FG optimization algorithm and the mentioned optimization algorithms, the normalized results were presented in Table 7. It can be seen from Table 7 that the proposed FG algorithm is superior to all the mentioned optimization algorithms. Although DE, TLBO and ABC algorithms achieved the best results over F1 function, the proposed FG algorithm obtained an acceptable result. DE algorithm achieved the best result over function F2 and the proposed FG algorithm achieved the second rank. It can be also observed from Table 7 that the proposed FG method achieved the highest performance over the functions F4, F6, F8, F10, F14, F16, F18, F19 and F20 and obtained acceptable results for the remaining functions.

Table 6. The controlling parameters of the algorithms that participated in the experiment for unconstrained benchmark functions.

Algorithm	Controlling Parameters
KH	Number of krills: 25 Maximum number of iterations: 200 Inertia weights: 0.1–0.9 Foraging speed: 0.02 Maximum diffusion speed: 0.005 Maximum induced speed: 0.01
PSO	Population size: 50 Inertia weight: 0.6 Cognitive parameter: 1.65 Social parameter: 2
GA	Population size for low-dimensional functions: 50 Population size for high-dimensional functions: 200 Selection function: Roulette Crossover fraction: 0.8 (single-point) Elite count for replacement: 0.05
DE	Population size: 50 Crossover factor: 0.5 Constant factor: 0.5
ABC	Number of employed bees: 25 Number of onlooker bees: 25 Limit: Number of iterations
TLBO	Population size: 50
The proposed FG algorithm	Participants number: 20 Impact coefficient: Based on the fitness value Inertia weight: 0.8 Replacement operator: 0.05

4.4.2. *Experiments on constrained benchmark functions*

In this sub-subsection, we evaluate the performance of the proposed FG optimization algorithm using the mentioned 24 constrained benchmark functions. In order to investigate the performance of the proposed FG algorithm, the results obtained by the proposed FG algorithm are compared with those of some well-known optimization algorithms such as PSO, biogeography-based optimization (BBO),[60] DE, ABC and HTS[26] which have been frequently used by different researchers.

The maximum function evaluation for each constrained benchmark function is considered 240,000. The controlling parameters of BBO, PSO, ABC, DE, TLBO, HTS and the proposed FG algorithms are set according to Table 8. Moreover, all the mentioned algorithms are tested with the static penalty method as a constraint-handling technique. It should be mentioned that in static penalty method, a penalty value is added to the fitness value of each infeasible individual so that it will be penalized for violating the constraints.

The obtained results of the proposed FG algorithm for the mentioned constrained benchmark functions in 50 trials with 200 iterations are presented in Table 9.

Table 7. The average values of the normalized results of the proposed FG optimization algorithm and several famous meta-heuristic optimization algorithms in 50 trials for the unconstrained benchmark functions.

ID	FG	KH	PSO	GA	DE	ABC	TLBO
F1	0.998	0.942	0.988	0	1	1	1
F2	0.914	0.877	0.952	0.350	1	0.906	0.852
F3	0.739	0.235	0.617	0.226	0.964	0.849	0.242
F4	1	0.937	0.379	0.687	0	0.381	0.691
F5	0.446	1	0	0.125	0.536	0.498	0.511
F6	1	0.999	0.988	0	0.999	0.999	0.999
F7	0.999	0.999	0.999	0	0.999	1	1
F8	1	0.999	0.999	0	0.999	1	1
F9	0.998	0.992	0.197	0	0.966	0.9921	1
F10	1	0.565	0.610	0.299	0	0.305	0.305
F11	0.896	0.378	1	0	0.777	0.805	0.805
F12	0.999	1	0.661	0	0.998	0.999	0.999
F13	0.655	0.655	0.6552	0	1	0.655	0.655
F14	1	0.999	0.502	0.998	0	0.927	0.930
F15	0.992	1	0.968	0	0.751	0.756	0.752
F16	1	0.999	0	0	0.99	0.999	0.989
F17	1	0.999	0.023	0	0.773	0.999	1
F18	1	0.999	0	0.999	0.999	1	1
F19	1	0.999	0.999	0	0.999	0.999	0.999
F20	1	1	0.999	0	0.999	1	1
AVG	0.932	0.879	0.627	0.184	0.787	0.853	0.836
Rank	1	2	6	7	5	3	4

Moreover, the normalized results obtained by the proposed FG algorithm and other mentioned meta-heuristic optimization algorithms for G1–G24 are tabulated in Table 10.

It can be seen from Table 9 that the proposed FG optimization algorithm performs well in finding the optimal solutions which confirm the high performance of the proposed FG optimization algorithm. The obtained results on 24 constrained benchmark functions shown in Table 10 prove the superiority of the proposed FG method over the other well-known optimization algorithms. It should be mentioned that HTS and ABC algorithms obtained the second and third ranks, respectively.

In the experiments over G1 benchmark function, FG, HTS and ABC algorithms achieved the best results and are superior to the other optimization algorithms. Although the BBO and FG algorithms achieved acceptable results in dealing with G2 test function, ABC algorithm achieved the best result and ranked first in this experiment. The experimental results on G3 show the superiority of the DE and ABC algorithms over the other mentioned optimization algorithms. However, the results achieved by HTS and FG algorithms are acceptable too. Moreover, in the experiments on G4 and G5, almost all algorithms except BBO achieved good results. ABC, FG, HTS, TLBO and DE algorithms performed well and achieved acceptable results

Table 8. The controlling parameters of the algorithms that participated in the experiment for unconstrained benchmark functions.

Algorithm	Controlling Parameters
PSO	Population size: 50 Inertia weight: 0.6 Cognitive parameter: 1.65 Social parameter: 2
ABC	Number of employed bees: 25 Number of onlooker bees: 25 Limit: Number of iterations
BBO	Population size: 50 Immigration rate: 1 Emigration rate: 1 Mutation factor: 0.01
DE	Population size: 50 Crossover factor: 0.5 Constant factor: 0.5
TLBO	Population size: 50
HTS	Population size: 50 Conduction factor: 2 Radiation factor: 2 Convection factor: 10
The proposed FG algorithm	Participants number: 20 Impact coefficient: Based on fitness value Inertia weight: 0.8 Replacement operator: 0.05

Table 9. The obtained results of the proposed FG algorithm for the constrained benchmark functions in 50 trials with 200 iterations.

ID	Best	Mean	Median	Worst	Std. Dev
G1	−15	−15	−15	−15	1.23E−18
G2	−0.803	−0.781	−0.767	−0.740	2.68E−02
G3	−1.0005	−0.891	−0.906	−0.384	6.55E−01
G4	−30665.538	−30665.538	−30665.538	−30665.538	1.08E−17
G5	5126.484	5136.803	5132.843	5268.825	1.69E+01
G6	−6961.81	−6961.81	−6961.81	−6961.81	2.18E−16
G7	24.2903	27.5119	27.1884	33.1847	1.45E+01
G8	0.095825	0.095825	0.095825	0.095825	1.26E−18
G9	680.6301	680.6301	680.6301	680.6301	1.48E−08
G10	7054.11	7229.561	7219.426	7726.13	1.87E+01
G11	0.7499	0.7499	0.7499	0.7499	0.9E−01
G12	−1	−1	−1	−1	1.53E−16
G13	0.217	0.3518	0.3124	0.6097	1.46E−01
G14	−47.1518	−46.3362	−46.0924	−45.5624	4.56E−01
G15	961.7150	973.51	970.849	988.5026	7.9E−01
G16	−1.9052	−1.9052	−1.9052	−1.9052	1.42E−16

Table 9. (*Continued*)

ID	Best	Mean	Median	Worst	Std. Dev
G17	8853.5396	8901.6423	8897.252	8915.232	1.91E+01
G18	−0.86603	−0.86602	−0.86602	−0.866	5.56E−07
G19	32.7326	32.9584	32.9214	33.1456	1.84E−01
G20	0.24583	0.2519	0.24916	0.27082	1.15E−02
G21	193.6503	204.5216	201.5973	240.3262	2.01E+01
G22	5.24E+06	4.89E+07	4.79E+07	6.28E+08	6.43E+06
G23	−394.7218	−192.1711	−206.2895	−18.8134	1.89+01
G24	−5.5080	−5.5080	−5.5080	−5.5080	8.15E−16

in dealing with G6 test function. DE optimization algorithm obtained a better performance in G7 test function. BBO optimization algorithm also found the best performance in dealing with G8 test function. All the optimization algorithms performed well over G9–G12 test functions. The best results on G13 and G14 test functions were achieved by FG optimization algorithm. The best performances in

Table 10. Comparative normalized results of the proposed FG optimization algorithm and different meta-heuristic optimization algorithms in 50 trials for G1–G24 constrained benchmark functions.

Function ID	PSO	DE	ABC	BBO	TLBO	HTS	FG
G1	0.931	0.894	1	0.945	0	1	1
G2	0	0.657	1	0.924	0.672	0.600	0.969
G3	0.610	1	1	0	0.669	0.835	0.819
G4	1	1	1	0	0.999	0.999	0.999
G5	0.990	0.862	0.941	0	0.999	1	0.989
G6	1	0.990	0.999	0	1	1	1
G7	0	1	0.979	0.318	0.934	0.977	0.604
G8	0	0	0	1	0	0	0
G9	1	1	0.999	0	0.999	0.999	0.999
G10	0.947	0.983	0.936	0	0.916	1	0.933
G11	1	0	0.993	0.463	0.994	0.994	0.994
G12	0	1	1	1	1	1	1
G13	0.706	0.298	0.168	0	0.571	0.542	1
G14	0.973	0.921	0.970	0	0.969	0.998	1
G15	0.905	0.999	0.885	0	0.971	1	0.691
G16	1	1	1	0	1	1	1
G17	0.954	0.890	0.871	0	1	0.997	0.999
G18	0.871	0.985	1	0	0.9993	0.679	0.933
G19	0.798	1	0.830	0	0.931	0.998	0.999
G20	0	0.999	0.994	0.987	0.990	1	1
G21	0.063	0.001	0.268	0	0.007	0.323	1
G22	0	0.999	0.999	1	0.999	1	1
G23	0.280	0.171	0.154	0	0.617	0.895	1
G24	1	1	1	0	1	1	1
AVG	0.626	0.777	0.833	0.276	0.802	0.867	0.913
Rank	6	5	3	7	4	2	1

Table 11. Friedman rank test for mean solutions obtained for unconstrained benchmark functions.

Algorithm	Friedman Value	Normalized Value	Rank
FG	59	1	1
KH	94	0.674	5
PSO	125	0.386	8
GA	166.5	0	9
DE	106.5	0.558	7
ABC	88.5	0.725	4
TLBO	86	0.748	3

Table 12. Friedman rank test for mean solutions obtained for constrained benchmark functions.

Algorithm	Friedman Value	Normalized Value	Rank
PSO	104.5	0.5	6
BBO	136.5	0	7
DE	93.5	0.671	5
ABC	92	0.695	3
TLBO	92.5	0.687	4
HTS	73.5	0.984	2
FG	72.5	1	1

dealing with G15 test function were obtained by DE and HTS optimization algorithms. It can be also seen from Table 10 that the TLBO, ABC, DE and HTS algorithms achieved the best results in dealing with G17, G18, G19 and G20 test functions, respectively, while the proposed FG algorithm showed acceptable performance in dealing with these test functions. Moreover, it can be observed from Table 10 that the proposed FG method found better performance on G21–G24 test functions in comparison with the other meta-heuristic optimization algorithms which participated in this experiment. The average of achieved results shows the superiority of the proposed FG algorithm over the other participating algorithms in this experiment.

Another useful statistical test to compare the real performance of the meta-heuristic optimization algorithms is Friedman rank test. The Friedman rank test is a multi-comparison test which detects the behavior differences between two or more algorithms. We conducted this experiment on the results achieved by the proposed FG algorithm and also the mentioned meta-heuristic optimization algorithms on the presented constrained and unconstrained benchmark functions. The results of this experiment are tabulated in Tables 11 and 12.

Values tabulated in Tables 11 and 12 are normalized Friedman values, achieved in Friedman tests. Here "0" indicates the worst and "1" indicates the best algorithms. The algorithms have been ranked based on normalized values.

Tables 11 and 12 indicate that the proposed FG achieved the best results and is ranked first.

5. Comparative Study

This section tends to highlight the main characteristics of the proposed FG algorithm and provides concise and compact comparison between the proposed algorithm and some famous evolutionary techniques such as BSO, TLBO and BBO algorithms.

5.1. *FG versus BSO*

FG and BSO both are swarm-intelligence-based algorithms which were inspired by the human behavior and cooperation in achieving solutions for the subjects they are dealing with. Although they are similar, they differ in some aspects. Some main similarities and differences are listed below:

- Main similarity between FG and BSO is their source of inspiration. Human cooperative behavior in working out solutions for issues they are dealing with is the inspiration source of these algorithms. However due to the different points of view for this similar inspiration source, completely different algorithms have been developed.
- Due to their different points of view, they employ different strategies and also different equations for searching the problem spaces. They also use different governing rules that result in different behaviors. This procedure seems like the procedure of evolution of the individual in evolutionary techniques while FG algorithm uses the classic definition of interaction between individuals (searching agents) to reach the optimal solution.
- Updating process of these algorithms differs in some ways. FG algorithm evaluates fitness of the solutions and then calculates the impact of each solution on the other solutions and tries to improve solutions considering the impacts of all solutions. On the other hand, BSO's main idea is to generate diverse solutions, classify them into some clusters and with generated probability values select individuals as well as clusters (and also through stochastic manner) to generate new individuals. Then, a selection procedure decides whether these generated individuals can replace chosen individuals or not.

5.2. *FG versus TLBO*

- TLBO algorithm is inspired by teaching–learning process based on the output of learners influenced by their teacher's impact while FG is inspired by the cooperation behavior of humans in sharing ideas on a subject and improving their solutions by discussion.

- Updating process of TLBO is done in two different phases, teaching phase and learning phase. In teaching phase, the mean result of the class is tried to increase based on teacher's capability while in learning phase, learners try to increase their knowledge by interacting with each other. In FG algorithm, updating process is done based on cooperation between group members, considering the impact of each solution on the other solutions. These algorithms also employ different operators and equations in their updating process.
- Both algorithms belong to swarm-intelligence-based algorithms. Interactions between members in these algorithms in different forms of idea sharing processes result in finding optimal solution for the problems.

5.3. *FG versus BBO*

- BBO algorithm is inspired by the biogeography, geographical distribution of biological spices, while FG algorithm is inspired by the human cooperative behavior.
- BBO belongs to the bioinspired algorithms but not swarm-intelligence-based methods meaning that it is inspired by the biological process but not using swarming behavior. However, FG algorithm belongs to the bioinspired and swarm-intelligence-based algorithms.
- Updating process in BBO is influenced by two main operators, migration and mutation. Migration operator identifies regions for habitation of the species based on their qualities. High-quality solutions are considered as habitats that can have large number of spices while low-quality ones serve small number of spices. FG algorithm uses defined rules of sharing and affecting solutions in its updating process.

6. Conclusion

In this paper, a new meta-heuristic algorithm namely *focus group* inspired by the cooperative behavior of a focus group in sharing the solutions of the members in order to obtain the best solution about a specific subject was introduced. Each participant proportional to its fitness has impact on the other participants' solutions. The solutions of participants are affected more by their solutions which have higher quality. In the proposed FG optimization algorithm, based on the mentioned mechanism, each solution is updated and the best solution is recorded by the note taker in every iteration. This iterative process is continued and finally the best solution is obtained. In order to evaluate the proposed FG algorithm, we examined it on a set of different well-known standard benchmark functions containing constrained and unconstrained functions. It is very encouraging to find out that the proposed FG optimization algorithm is superior to most of the compared meta-heuristic algorithms. To compare the real performances of the proposed FG optimization algorithm and the other meta-heuristic algorithms, Friedman rank tests were also utilized. This research reveals that the proposed FG optimization algorithm is a novel meta-heuristic optimization algorithm which can drastically deal with optimization problems.

Acknowledgments

This work was supported by Shahid Rajaee Teacher Training University under the Contract No. 26643.

Conflict of Interest

The authors declare no conflict of interest.

References

1. X.-S. Yang, *Nature-Inspired Meta-Heuristic Algorithms* (Luniver Press, 2010).
2. X.-S. Yang, *Engineering Optimization: An Introduction with Meta-Heuristic Applications* (John Wiley & Sons, 2010).
3. S. Kirkpatrick, Scott, C. D. Gelatt, Jr. and M. P. Vecchi, Optimization by simulated annealing, *Science* **220**(4598) (1983) 671–680.
4. K. Sörensen, Metaheuristics — The metaphor exposed, *Int. Trans. Oper. Res.* **22**(1) (2015) 3–18.
5. G. E. Goldberg and J. H. Holland, Genetic algorithms and machine learning, *Mach. Learn.* **3**(2) (1988) 95–99.
6. M. Mitchell, *An Introduction to Genetic Algorithms* (MIT Press, 1998).
7. R. Storn and K. Price, Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces, *J. Glob. Optim.* **11**(4) (1997) 341–359.
8. L. J. Fogel, A. J. Owens and M. J. Walsh, *Artificial Intelligence through Simulated Evolution* (Wiley-IEEE Press, 2009).
9. Z. W. Geem, J. H. Kim and G. V. Loganathan, A new heuristic optimization algorithm: Harmony search, *Simulation* **76**(2) (2001) 60–68.
10. J. Kennedy, Particle swarm optimization, in *Encyclopedia of Machine Learning* (Springer US, 2010), pp. 760–766.
11. M. Dorigo, V. Maniezzo and A. Colorni, Ant system: Optimization by a colony of cooperating agents, *IEEE Trans. Syst. Man Cybern. B, Cybern.* **26**(1) (1996) 29–41.
12. D. Karaboga and B. Basturk, A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm, *J. Glob. Optim.* **39**(3) (2007) 459–471.
13. X.-S. Yang, Firefly algorithm, stochastic test functions and design optimization, *Int. J. Bio-Inspir. Comput.* **2**(2) (2010) 78–84.
14. R. Rajabioun, Cuckoo optimization algorithm, *Appl. Soft Comput.* **11**(8) (2011) 5508–5518.
15. E. Rashedi, H. Nezamabadi-Pour and S. Saryazdi, GSA: A gravitational search algorithm, *Inf. Sci.* **179**(13) (2009) 2232–2248.
16. H. Shah-Hosseini, The intelligent water drops algorithm: A nature-inspired swarm-based optimization algorithm, *Int. J. Bio-Inspir. Comput.* **1**(1–2) (2009) 71–79.

17. Y. Shi, Brain storm optimization algorithm, in *Proc. Int. Conf. Swarm Intelligence* (Springer, Berlin, 2011).
18. A. H. Gandomi and A. H. Alavi, Krill herd: A new bio-inspired optimization algorithm, *Commun. Nonlinear Sci. Numer. Simul.* **17**(12) (2012) 4831–4845.
19. A. Kaveh and N. Farhoudi, A new optimization method: Dolphin echolocation, *Adv. Eng. Softw.* **59** (2013) 53–70.
20. C. Sur, S. Sharma and A. Shukla, Egyptian vulture optimization algorithm: A new nature inspired meta-heuristics for knapsack problem, in *Proc. 9th Int. Conf. Computing and Information Technology (IC2IT2013)* (Springer, Berlin, 2013).
21. S. Mirjalili, S. M. Mirjalili and A. Lewis, Grey wolf optimizer, *Adv. Eng. Softw.* **69** (2014) 46–61.
22. S. Mirjalili, The ant lion optimizer, *Adv. Eng. Softw.* **83** (2015) 80–98.
23. Y.-J. Zheng, Water wave optimization: A new nature-inspired meta-heuristic, *Comput. Oper. Res.* **55** (2015) 1–11.
24. A. H. Kashan, A new meta-heuristic for optimization: Optics inspired optimization (OIO), *Comput. Oper. Res.* **55** (2015) 99–125.
25. A. H. Gandomi, Interior search algorithm (ISA): A novel approach for global optimization, *ISA Trans.* **53**(4) (2014) 1168–1183.
26. V. K. Patel and V. J. Savsani, Heat transfer search (HTS): A novel optimization algorithm, *Inf. Sci.* **324** (2015) 217–246.
27. S. Mirjalili, Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems, *Neural Comput. Appl.* **27**(4) (2016) 1053–1073.
28. S. Mirjalili, S. M. Mirjalili and A. Hatamlou, Multi-verse optimizer: A nature-inspired algorithm for global optimization, *Neural Comput. Appl.* **27**(2) (2016) 495–513.
29. W. Du and B. Li, Multi-strategy ensemble particle swarm optimization for dynamic optimization, *Inf. Sci.* **178**(15) (2008) 3096–3109.
30. X. Yao, Y. Liu and G. Lin, Evolutionary programming made faster, *IEEE Trans. Evol. Comput.* **3**(2) (1999) 82–102.
31. V. Chellaboina and M. K. Ranga, Reduced order optimal control using genetic algorithms, in *Proc. 2005 American Control Conf.* (IEEE, 2005).
32. H. H. Balci and J. F. Valenzuela, Scheduling electric power generators using particle swarm optimization combined with the lagrangian relaxation method, *Int. J. Appl. Math. Comput. Sci.* **14**(3) (2004) 411–422.
33. B. Zhao and S. Li, Ant colony optimization algorithm and its application to neuro-fuzzy controller design, *J. Syst. Eng. Electron.* **18**(3) (2007) 603–610.
34. R. L. Johnston and H. M. Cartwright (eds.), *Applications of Evolutionary Computation in Chemistry*, Structure and Bonding, Vol. 110 (Springer Science & Business Media, 2004).
35. C.-C. Wu, K.-C. Lai and R.-Y. Sun, GA-based job scheduling strategies for fault tolerant grid systems, in *Proc. Asia-Pacific Services Computing Conf. (APSCC'08)* (IEEE, 2008).
36. Y. Liu *et al.*, A tabu search approach for the minimum sum-of-squares clustering problem, *Inf. Sci.* **178**(12) (2008) 2680–2704.
37. X. Tan and B. Bhanu, Fingerprint matching by genetic algorithms, *Pattern Recognit.* **39**(3) (2006) 465–477.
38. H. R. Kanan and K. Faez, GA-based optimal selection of PZMI features for face recognition, *Appl. Math. Comput.* **205**(2) (2008) 706–715.
39. H. R. Kanan and K. Faez, An improved feature selection method based on ant colony optimization (ACO) evaluated on face recognition system, *Appl. Math. Comput.* **205**(2) (2008) 716–725.

40. H. R. Kanan, K. Faez and M. Hosseinzadeh, Face recognition system using ant colony optimization-based selected features, in *Proc. 2007 IEEE Symp. Computational Intelligence in Security and Defense Applications* (IEEE, 2007).
41. H. E. Kanan, K. Faez and S. M. Taheri, Feature selection using ant colony optimization (ACO): A new method and comparative study in the application of face recognition system, in *Proc. ICDM 2007: Industrial Conf. Data Mining* (Springer, Berlin, 2007).
42. O. Cordón, S. Damas and J. Santamaría, A fast and accurate approach for 3D image registration using the scatter search evolutionary algorithm, *Pattern Recognit. Lett.* **27**(11) (2006) 1191–1200.
43. H. Nezamabadi-Pour, S. Saryazdi and E. Rashedi, Edge detection using ant algorithms, *Soft Comput.* **10**(7) (2006) 623–628.
44. H. A. Kanan, K. Faez and M. Ezoji, An efficient face recognition system using a new optimized localization method, in *Proc. 18th Int. Conf. Pattern Recognition (ICPR'06)*, Vol. 3 (IEEE, 2006).
45. H. R. Kanan and B. Nazeri, A novel image steganography scheme with high embedding capacity and tunable visual image quality based on a genetic algorithm, *Expert Syst. Appl.* **41**(14) (2014) 6123–6130.
46. H. R. Kanan and M. H. Moradi, A genetic algorithm based method for face localization and pose estimation, in *Proc. 3rd Int. Conf. Sciences of Electronic, Technologies of Information and Telecommunications* (2005).
47. H. R. Kanan and K. Faez, ZMI and wavelet transform features and SVM classifier in the optimized face recognition system, in *Proc. Fifth IEEE Int. Symp. Signal Processing and Information Technology* (IEEE, 2005).
48. H. R. Kanan and K. Faez, PZMI and wavelet transform features in face recognition system using a new localization method, in *Proc. 31st Annu. Conf. Industrial Electronics Society (IECON 2005)* (IEEE, 2005).
49. E. Khoshkerdar and H. R. Kanan, Gender classification using GA-based adjusted order PZM and fuzzy similarity measure, in *Proc. 2013 13th Iranian Conf. Fuzzy Systems (IFSC)* (IEEE, 2013).
50. S. Nazari, M.-S. Moin and H. R. Kanan, A discriminant binarization transform using genetic algorithm and error-correcting output code for face template protection, *Int. J. Mach. Learn. Cybern.* (2017), doi: 10.1007/s13042-017-0723-3.
51. M. Ezoji *et al.*, GA-based affine PPM using matrix polar decomposition, *IEICE Trans. Inf. Syst.* **89**(7) (2006) 2053–2060.
52. D. H. Wolpert and W. G. Macready, No free lunch theorems for optimization, *IEEE Trans. Evol. Comput.* **1**(1) (1997) 67–82.
53. E.-G. Talbi, *Metaheuristics: From Design to Implementation*, Wiley Series on Parallel and Distributed Computing, Vol. 74 (John Wiley & Sons, 2009).
54. Focus Groups: Background and "How To" Guidelines, 1995.
55. J. J. Liang *et al.*, Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization, *J. Appl. Mech.* **41** (2006) 8.
56. E. Mezura-Montes and C. A. C. Coello, A simple multimembered evolution strategy to solve constrained optimization problems, *IEEE Trans. Evol. Comput.* **9**(1) (2005) 1–17.
57. A. E. Muñoz Zavala, A. Hernández Aguirre and E. R. Villa Diharce, Constrained optimization via particle evolutionary swarm optimization algorithm (PESO), in *Proc. 7th Annu. Conf. Genetic and Evolutionary Computation* (ACM, 2005).

58. R. V. Rao, V. J. Savsani and D. P. Vakharia, Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems, *Comput. Aided Des.* **43**(3) (2011) 303–315.
59. R. V. Rao, V. J. Savsani and D. P. Vakharia, Teaching–learning-based optimization: An optimization method for continuous non-linear large scale problems, *Inf. Sci.* **183**(1) (2012) 1–15.
60. D. Simon, Biogeography-based optimization, *IEEE Trans. Evol. Comput.* **12**(6) (2008) 702–713.