Available online at www.sciencedirect.com

ScienceDirect

Future Computing and Informatics Journal xx (2018) 1–9
http://www.journals.elsevier.com/future-computing-and-informatics-journal/

TGA: Team game algorithm

M.J. Mahmoodabadi [a,*], M. Rasekh [a], T. Zohari [b]

[a] Department of Mechanical Engineering, Sirjan University of Technology, Sirjan, Iran
[b] Department of Mechanical Engineering, University of Politecnico di Milano, Milan, Italy

Received 12 December 2017; accepted 30 March 2018
Available online ■ ■ ■

Abstract

Lately, there is a growing interest in conducting research on optimization algorithms due to their wide range of engineering applications. One of the optimization algorithms' categories is evolutionary algorithms which are inspired from the natural behavior of animals and humans. Further, each of the evolutionary algorithms has its own advantages and disadvantages in convergence accuracy and computational time. In the present paper, a novel solution search algorithm taken from the team games is introduced. This evolutionary algorithm named Team Game Algorithm (TGA) involves passing a ball, making mistakes and substitution operators. Comparing the TGA's results to the outcomes of other well-known algorithms for unimodal and multimodal test functions elucidates the successful design of the proposed heuristic algorithm.

Keywords: Optimization heuristic algorithm; Team game algorithm; Passing operator; Mistake operator; Substitution operator; Unimodal and multimodal test functions

1. Introduction

The beginning of the 20th century could be considered as an inception of the extensive use of mathematical models and optimization fields. Optimization consists of numerous fields, such as operations research, the artificial intelligence and even computer sciences. Therefore, these fields together could help us to improve the efficiency of industries especially the practical ones. In fact, we need to resolve particular problems that couldn't be solved by traditional methods. In this respect, evolutionary computations as one of the sub-categories of artificial intelligence have been utilized to address those particular problems. This idea is based on a multi-point search instead of one-point originally introduced by Rechenberg [24] and [23]. Later, Holland introduced the Genetic Algorithm (GA) as the first evolutionary algorithm for the global optimization using chromosomes and their genes [10]. Then, [4], introduced the Ant Colony Optimization (ACO) algorithm inspired from the movements of ant. This algorithm performs based upon two principal rules: The first one is the footprint pheromone and the second one is the search of the path with more pheromones. Furthermore, Eberhart and Kennedy presented particle swarm optimization (PSO) [5] as another evolutionary algorithm which was based on the swarm behavior of birds and fishes. Indeed, PSO sets the position of a particle by using its best experience and the best experience of all particles [12]. introduced a new group-based algorithm, which imitates the foraging behavior of honeybees. Later, [1], utilized the political, social and economic behavior of countries to introduce novel operators. They presented an effective algorithm named Imperialist Competitive Algorithm (ICA). In their method, empires compete against each other and as the algorithm goes forward, the weakest empire loses its colonies [22]. used the theory of Newtonian physics and constructed a gravitational search algorithm (GSA) which was based on the law of the gravity and the notion of mass interactions.

These algorithms were some prominent evolutionary algorithms while many other algorithms have been introduced in the recent years, such as: Cuckoo Optimization Algorithm (COA)

* Corresponding author.
E-mail address: mahmoodabadi@sirjantech.ac.ir (M.J. Mahmoodabadi).
Peer review under responsibility of Faculty of Computers and Information Technology, Future University in Egypt.

https://doi.org/10.1016/j.fcij.2018.03.002

[31], Firefly Algorithm (FA) [30], Glowworm Swarm Optimization (GSO) [15], Bat Algorithm (BA) [32], Cat Swarm Optimization (CSO) [3], Fruit Fly Optimization Algorithm (FFOA) [19,29], Differential Evolution (DE) [25], Artificial Immune System (AIS) [11], Krill Herd (KH) [6], Levy-Flight Krill Herd (LKH) [28], Bacterial Foraging Optimization (BFO) [20], Lion Pride Optimization (LPO) [27], the lion's algorithm [21], Hunting Search Optimization (HUS) [18], Artificial Fish Swarm (AFS) [16], Simulated Annealing (SA) [14], Magnetic Optimization Algorithm (MOA) [17,26], Harmony Search (HS) [7], Probabilistic Hill-climbing Algorithm [9], etc.

In this paper, we're going to introduce a novel meta-heuristic optimization algorithm based on team game strategies involving football, volleyball, basketball, water polo and so on. Team games have to follow several straightforward rules which are vital for them to succeed. Certainly, one of the most essential strategies of team games is passing a ball, whereas players must have teamwork for achieving points or scoring goals. Another strategy of the team games is using the mistakes of the team's opponent to receive scores. In a football game, for instance, a mistake in the play of the team's opponent could lead to a powerful counter strike and an achievement of a goal. We also know that each team game has a playground territory in which players can play a role in and out of the playground territory. In a game, there may be an injured or a tired player whom must be substituted. These substitutions can speed up the pace of the game and even sometimes cause the team to score its goal. We have simulated these processes based on the team games' acts via regarding them as the optimization operators. The players are the population in the specified algorithm, which are divided into two groups; original and substituting players. In the proposed algorithm, passing a ball, making mistakes and substitution operators are simulated by mathematical formula. Furthermore, for the operators, we will check that the members are doing their activities in the defined territory appropriately.

The rest of this paper is organized as follows. In Section 2, a brief description of the team games and the rules are presented. Further, the strategy of the team game algorithm and the formulation of the operators are provided in Section 3. In Section 4, the results of the specified algorithm and the graphs are presented. Finally, Section 5 concludes the present study.

2. Team games

Generally speaking, games are divided into two categories. The first one is the individual games and the second one which we'll explicate here is the team games.

A team is defined as a number of members who have the same goals, same destinations and complementary skills. Based on these characteristics, they are bound to each other and relied upon one another. Katzenbach and Smith say: “members of a team are dependent together; it means no activity from a member, results in teamwork failure.” [13].

In the team games, there are two teams to compete; and in order to have a fair game, each team has the same number of players. In every team game, all the team players cooperate with each other. Moreover, the consequences affected the team and as a result affected all the players of that team. We will plumb this subject and analyze the team games which played with a ball. In this kind of games, both teams have the equal chance of obtaining the ball and starting the game. Furthermore, there is a field which is defined for two teams' players and they have the permission to move the ball just within that field. If the ball goes out of the specified field by the action of one of the players, referee declares that “the ball is out” and the ball will be given to the opponent team. Another obvious issue in these games is the substitution of the players. The players must be active and participate with their team members to succeed in the game although any unpredictable event could happen in the game. Most of these events are limited to fatigue and sometimes injuries. As a matter of fact, an injured or a fatigue player could no longer moves in the playground and must be substituted. In addition, we know that at anywhere and anytime of a game, the substitution could be performed and a fresh player can come in.

3. Team game algorithm (TGA)

In this section, a heuristic algorithm which stemmed from team games is proposed and is named Team Game Algorithm (TGA). In this algorithm, agents are players and their performance will be measured by their stamina which is called the propriety in the optimization methods. All players, which include the two teams’ players, have contacts with each other. Cooperation between teammates and involvements against the players of the opponent team result in all players try to catch up the goals. Players with higher stamina stay more in field, even sometimes to the end of the game.

In TGA, each player (agent) has three kinds of operations: passing a ball, making mistakes and substitution. Passing a ball is a logical operator and mistaking mistakes is a stochastic operator. Also, substitution happens in the tired team. By passing balls, we expect that a team wins the game and a player who is the best of that match is introduced. This player could even belong to the loser team and is the optimum point in the specified search space.

3.1. Initialization

Consider a system having n agents (players). First of all we create n players randomly in the search space. Half of the players will be chosen to play in team A and the rest of them will be given to team B. The number of substituting players is free. They will sit on the football substitute bench and wait for a coach order to be substituted.

$$X = \begin{pmatrix} x_1^1 & x_2^1 & \cdots & x_n^1 \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & \vdots & \vdots & \vdots \\ x_1^D & x_2^D & \cdots & x_n^D \end{pmatrix} = (x_1, x_2, \cdots, x_n) \tag{1}$$

where x_n^D is the position of the nth player in the direction of D (dimension). Also, the number of all players (n) contains the summation of players in teams A and B.

Please cite this article in press as: Mahmoodabadi MJ, et al., TGA: Team game algorithm, Future Computing and Informatics Journal (2018), https://doi.org/10.1016/j.fcij.2018.03.002

$$nA = nB = \frac{n}{2} \tag{2}$$

In Equation (2), nA and nB are the number of players in teams A and B, respectively, who play on the field.

Now each team selects its own players from the existing players (X) and sets up its team.

$$A = (x_{1A}, x_{2A}, \cdots, x_{nA}) \tag{3}$$

$$B = (x_{1B}, x_{2B}, \cdots, x_{nB}) \tag{4}$$

Moreover,

$$A \cup B = X \tag{5}$$

where $\cup$ is the union operator.

After all, the ball will be given to a team randomly and three operators will be applied on that team with predetermined probabilities.

3.2. Passing operator

The first operator is passing a ball, and is defined as follows:

$$x_i(t+1) = x_i(t) + r \times C_i(t+1) \tag{6}$$

$$C_i(t+1) = 2x_{cap}(t) - x_i(t) - x_{rand}(t) \tag{7}$$

where $x_i(t)$ is the position of the ith player in the host team. C is a communicative formula between the specified player who has the ball, a random player $x_{rand}(t)$ and the captain of the team $x_{cap}(t)$. Also in the above equation, r is a random vector that the values of its components are between [0,1]. The schematic illustrations of Equations (6) and (7) are provided in Figs. 1 and 2.

3.3. Mistake operator

The second operator is the players' mistake and will be performed when its probability condition is satisfied. In this operator, a player of the ball-owner team and his/her peer in the opponent team contact with each other. This contact results in changing a random dimension of the player having the ball via the following equation.

$$x_i^d(t+1) = x_i^d(t) + z \times \left(x_j^d(t) - x_i^d(t)\right) \tag{8}$$

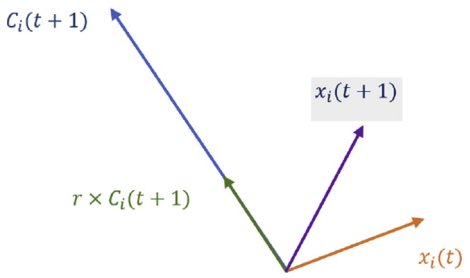

Fig. 1. The schematic illustration of the Equation (6).

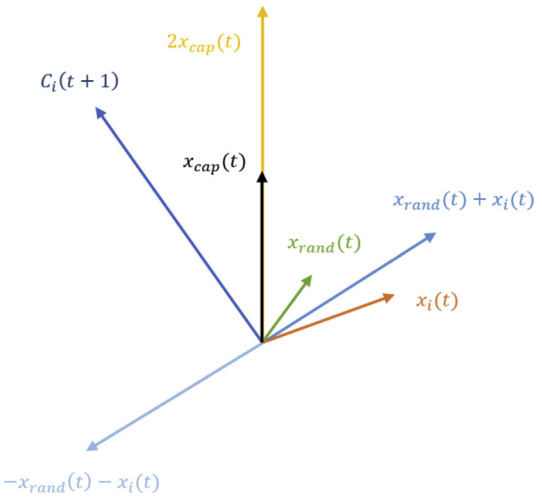

Fig. 2. The schematic illustration of the Equation (7).

where $x_i^d(t)$ is the position of the ith player with its random dimension d. Also, z is a random value between [−1,+1]. Furthermore, $x_j^d(t)$ represents the position of the randomly selected jth player of team B with dimension d.

3.4. Substitution operator

For the third operator, w e will check the player's propriety during the iterations; if that player is unable to improve his propriety in the specified iterations, he will be substituted and a fresh player with a random position enters the game. Further, it must be mentioned that not only improper but also constant proprieties for a player during the iterations could lead to a substitution, although a player with a worsening propriety might occur less in the game.

3.5. Out of the field players

At the end of all these operations, the position of the ball-owner player will be checked. If he has gone out of the field, the player's location will be reset and he will be given a new random position in the specified field.

With a general view, the flowchart of TGA is shown in Fig. 3. For a clear vision of the structure of TGA, some notes must be regarded:

(1) Players in TGA could observe each other's performance; hence it causes TGA not to be a blind search algorithm.
(2) The players will learn from their mistakes to improve their propriety.
(3) The substitution operator will prevent the algorithm from getting stuck in the local minima.

In Fig. 4, a pseudo code for the proposed algorithm is shown. In this pseudo code, p_p is passing probability and *game time* is the total number of iterations. Also x_{min} and x_{max} specify the minimum and maximum domain of the playground which is different in every function. $tr(i)$ is the times which a player doesn't give a better propriety in the sequential iterations.

Please cite this article in press as: Mahmoodabadi MJ, et al., TGA: Team game algorithm, Future Computing and Informatics Journal (2018), https://doi.org/10.1016/j.fcij.2018.03.002

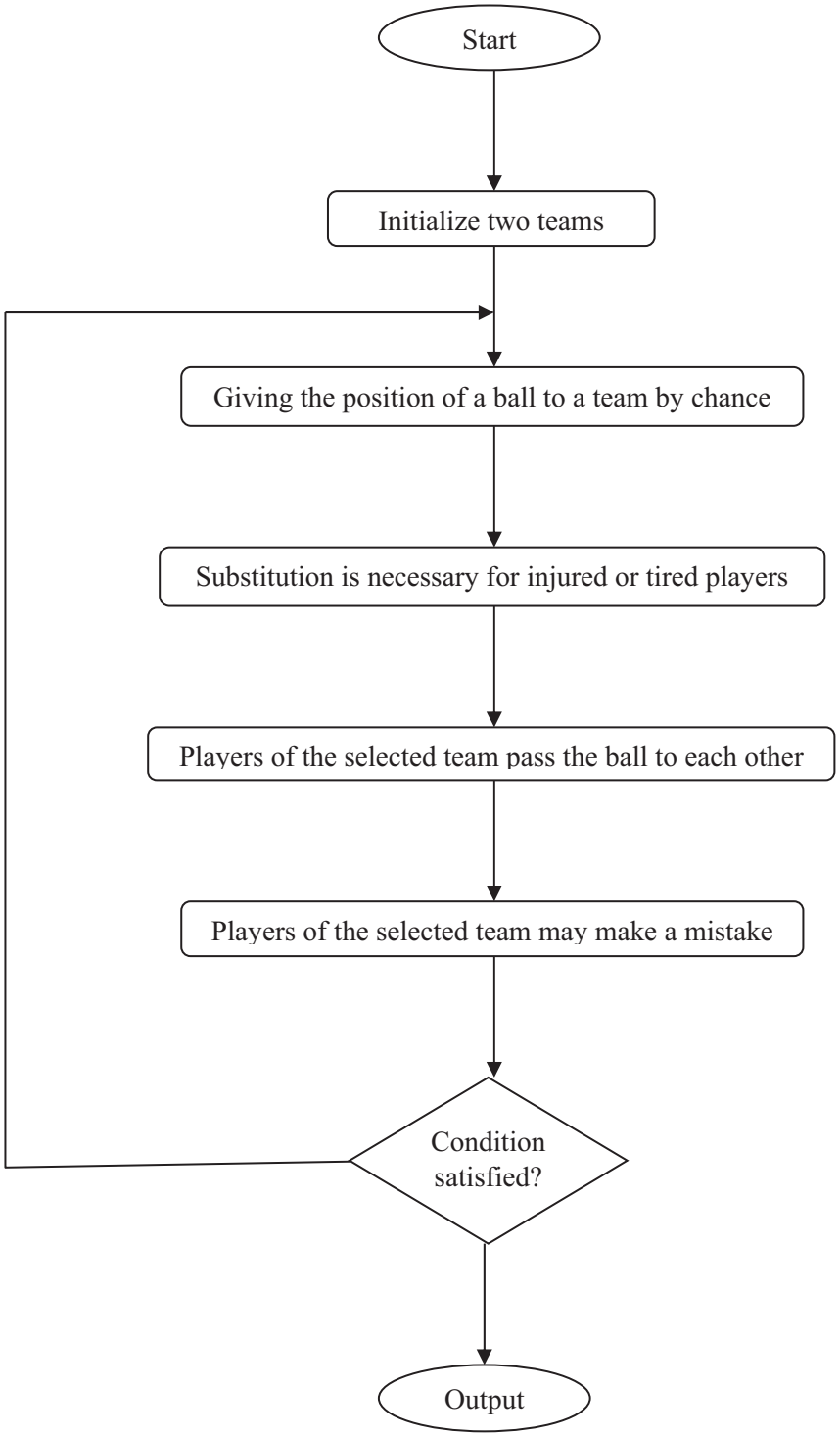

Fig. 3. A general flowchart of TGA.

4. Numerical and comparative results

In this section, the proposed algorithm has been tested by 10 standard test functions; and also, the results are compared to the outcomes of three known algorithms in order to illustrate the performance of TGA. In Tables 1 and 2, unimodal and multimodal test functions with their names, mathematical formulations and their domains have been listed. Unimodal test functions don't have any local minima while multimodal test functions have one or more local minima. The minimum values of test functions in both Tables 1 and 2 are zero. Furthermore, these functions are minimized when $x_i = 0; \quad i = 1, 2, \ldots, n$ except for the function f5, which its minimum occurs in $x_i = 1; \quad i = 1, 2, \ldots, n$ where n is the maximum number of dimensions.

Please cite this article in press as: Mahmoodabadi MJ, et al., TGA: Team game algorithm, Future Computing and Informatics Journal (2018), https://doi.org/10.1016/j.fcij.2018.03.002

Initialize the positions and proprieties of players in two teams
set p_p
For t= 1: game time
If rand≤ 0.5
Set team A as the owner-ball team
Else
Set team B as the owner-ball team
End
For i= 1: the number of the players of a team
If the fitness of ith player hasn't been better
$tr(i) = tr(i) + 1$
End
if tr(i)==limit number
Replace $x_i(t)$*by a random new player*
Else if rand≤ p_p
$C_i(t+1) = 2x_{cap}(t) - x_i(t) - x_{rand}(t)$
$x_i(t+1) = x_i(t) + r \times C_i(t+1)$
Else
$x_i^d(t+1) = x_i^d(t) + z \times (x_j^d(t) - x_i^d(t))$
End
if $O_i^{(t)} < x_{min} \parallel O_i^{(t)} > x_{max}$
Replace $O_i^{(t)}$ *by a random new player*
End
End
End

Fig. 4. The pseudo code of the team game algorithm.

Table 1
Unimodal test functions.

Function name	Formulation	Domain
f1: Sphere	$\sum_{i=1}^{n} x_i^2$	$[-100, 100]^n$
f2: Schwefel 2.22	$\sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10, 10]^n$
f3: Quadric	$\sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	$[-100, 100]^n$
f4: Schwefel 2.21	$\max_i \{\lvert x_i \rvert, 1 \le i \le n\}$	$[-100, 100]^n$
f5: Rosenbrock	$\sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-30, 30]^n$
f6: Step	$\sum_{i=1}^{n} (\lvert x_i + 0.5 \rvert)^2$	$[-100, 100]^n$
f7: Quadric noise	$\sum_{i=1}^{n} i x_i^4 + random[0, 1]$	$[-1.28, 1.28]^n$

Numerical results of these test functions for the proposed algorithm, Genetic Algorithm with Traditional-Crossover (GATC) [8], Genetic Algorithm with Multiple-Crossover (GAMC) [2], and Gravitational Search Algorithm (GSA) [22] are shown for 30 runs in Tables 3 and 4 for the maximum iteration of 4000 and in Tables 5 and 6 for the maximum iteration of 10,000. In all of these results, the population size is set to 50 and the dimension size is set to 30. For GA with either traditional-crossover or multiple-crossover, regeneration, crossover and mutation probabilities are set to 0.4, 0.5 and 0.1, respectively. Moreover, the mutation parameter of the traditional and multiple-crossover equals 1×10^{-5} and 1×10^{-10}, respectively. Further, for GSA, G_0 and α are set to 100 and 20 in sequence and K_0 decreases from 50 to 1, linearly. For the TGA, the probability of doing a passing operation is computed by Equation (9).

$$p_p = 1 - 0.1\left(\frac{t}{time}\right) \tag{9}$$

Also, *tr(i),* the substitution conditions for both teams are set to 2000.

For more details of how TGA's operators work, its graphs for functions f1, f5, f7, f8 and f9 have been plotted in Figs. 5–9, in sequence. In these graphs, the y axis is scaled as logarithmic. In TGA's graphs, the speed of convergence is

Table 2
Multimodal test functions.

Function name	Formulation	Domain
f8: Rastrigin	$\sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12, 5.12]^n$
f9: Ackley	$-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	$[-32, 32]^n$
f10: Griewank	$\frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^n$

Please cite this article in press as: Mahmoodabadi MJ, et al., TGA: Team game algorithm, Future Computing and Informatics Journal (2018), https://doi.org/10.1016/j.fcij.2018.03.002

Table 3
The results of comparing three algorithms (GATC, GAMC and GSA) to TGA for the minimization of test functions in Table 1 with 4000 iterations.

		GATC	GAMC	GSA	TGA
f1	Min	8.95×10^{-19}	3.61×10^{-25}	4.54×10^{-18}	**2.16×10^{-68}**
	Max	4.45×10^{-14}	1.89×10^{-18}	1.74×10^{-17}	**8.44×10^{-62}**
	Mean	2.09×10^{-19}	7.97×10^{-20}	1.00×10^{-17}	**2.88×10^{-65}**
	Median	1.94×10^{-16}	1.53×10^{-23}	9.50×10^{-18}	**1.01×10^{-65}**
	Std.dev.	8.07×10^{-15}	3.48×10^{-19}	2.90×10^{-18}	**1.54×10^{-62}**
f2	Min	9.67×10^{-11}	2.18×10^{-14}	1.17×10^{-8}	**9.44×10^{-36}**
	Max	9.02×10^{-8}	7.51×10^{-11}	2.45×10^{-8}	**2.79×10^{-32}**
	Mean	8.22×10^{-9}	4.74×10^{-12}	1.63×10^{-8}	**1.45×10^{-33}**
	Median	3.46×10^{-9}	5.57×10^{-13}	1.63×10^{-8}	**3.04×10^{-34}**
	Std.dev.	1.68×10^{-8}	1.53×10^{-11}	2.82×10^{-9}	**5.03×10^{-33}**
f3	Min	1.23	0.17	0.61	**6.45×10^{-11}**
	Max	289.09	48.33	9.38	**1.62×10^{-8}**
	Mean	37.11	4.82	2.75	**3.19×10^{-9}**
	Median	14.35	1.29	2.20	**1.51×10^{-9}**
	Std.dev.	65.41	9.28	1.96	**3.91×10^{-9}**
f4	Min	16.06	41.83	**1.44×10^{-9}**	7.15×10^{-4}
	Max	42.44	73.93	**2.93×10^{-9}**	2.07×10^{-2}
	Mean	32.26	55.40	**1.93×10^{-9}**	6.17×10^{-3}
	Median	33.05	54.32	**1.86×10^{-9}**	5.20×10^{-3}
	Std.dev.	6.21	7.93	**3.21×10^{-10}**	5.07×10^{-3}
f5	Min	1.18	2.23×10^{-2}	21.27	**1.59×10^{-6}**
	Max	339.35	315.52	22.21	**12.09**
	Mean	87.28	75.39	21.73	**1.98**
	Median	76.88	69.21	21.72	**0.13**
	Std.dev.	76.06	81.86	**0.21**	3.08
f6	Min	**0**	**0**	**0**	**0**
	Max	486	16	**0**	1
	Mean	85.36	1.73	**0**	3.33×10^{-2}
	Median	12.50	0	**0**	**0**
	Std.dev.	140.84	3.87	**0**	0.18
f7	Min	8.10×10^{-3}	3.10×10^{-2}	5.90×10^{-3}	**3.86×10^{-3}**
	Max	2.37×10^{-2}	7.14×10^{-2}	2.25×10^{-2}	**1.36×10^{-2}**
	Mean	1.54×10^{-2}	4.82×10^{-2}	1.19×10^{-2}	**7.98×10^{-2}**
	Median	1.36×10^{-2}	4.77×10^{-2}	1.14×10^{-2}	**7.91×10^{-3}**
	Std.dev.	4.40×10^{-3}	1.06×10^{-2}	3.60×10^{-3}	**2.46×10^{-3}**

Note: the bold values show the best results.

Table 4
The results of comparing three algorithms (GATC, GAMC and GSA) to TGA for the minimization of test functions in Table 2 with 4000 iterations.

		GATC	GAMC	GSA	TGA
f8	Min	29.84	57.70	9.94	**0.99**
	Max	104.47	221.87	21.89	**19.90**
	Mean	51.80	112.62	14.36	**6.64**
	Median	47.26	105.46	13.93	**5.97**
	Std.dev.	17.65	33.29	**2.76**	3.95
f9	Min	5.92	8.90	1.97×10^{-9}	**7.99×10^{-15}**
	Max	14.44	17.36	**3.37×10^{-9}**	0.13
	Mean	10.49	14.28	**2.54×10^{-9}**	1.65×10^{-2}
	Median	10.51	14.77	2.48×10^{-9}	**1.68×10^{-14}**
	Std.dev.	1.91	2.03	**3.68×10^{-10}**	3.05×10^{-2}
f10	Min	7.40×10^{-3}	**0**	**0**	**0**
	Max	10.07	2.35	0.24	**2.46×10^{-2}**
	Mean	0.87	0.23	1.22×10^{-2}	**6.97×10^{-3}**
	Median	0.19	0.11	**0**	5.26×10^{-8}
	Std.dev.	1.98	0.45	4.52×10^{-2}	**9.18×10^{-3}**

Note: the bold values show the best results.

Table 5
The results of comparing three algorithms (GATC, GAMC and GSA) to TGA for the minimization of test functions in Table 1 with 10000 iterations.

		GATC	GAMC	GSA	TGA
f1	Min	8.95×10^{-28}	9.90×10^{-46}	4.16×10^{-18}	**5.10×10^{-74}**
	Max	1.65×10^{-22}	1.83×10^{-28}	1.16×10^{-17}	**6.17×10^{-65}**
	Mean	1.00×10^{-23}	7.18×10^{-30}	7.48×10^{-18}	**2.08×10^{-66}**
	Median	6.16×10^{-25}	8.71×10^{-34}	7.65×10^{-18}	**1.78×10^{-70}**
	Std.dev.	3.23×10^{-23}	3.34×10^{-29}	1.90×10^{-18}	**1.12×10^{-65}**
f2	Min	5.84×10^{-15}	1.22×10^{-23}	1.05×10^{-8}	**2.67×10^{-38}**
	Max	1.28×10^{-11}	7.57×10^{-14}	2.45×10^{-8}	**7.53×10^{-34}**
	Mean	1.68×10^{-12}	4.82×10^{-15}	1.28×10^{-8}	**6.81×10^{-35}**
	Median	1.13×10^{-13}	8.54×10^{-17}	1.27×10^{-8}	**1.05×10^{-35}**
	Std.dev.	3.32×10^{-12}	1.49×10^{-14}	1.25×10^{-9}	**1.54×10^{-34}**
f3	Min	9.51×10^{-8}	6.68×10^{-9}	**1.97×10^{-17}**	3.18×10^{-17}
	Max	2.60×10^{-3}	4.38×10^{-4}	**6.66×10^{-17}**	6.60×10^{-14}
	Mean	2.39×10^{-4}	2.13×10^{-5}	**3.66×10^{-17}**	5.90×10^{-15}
	Median	2.80×10^{-5}	2.17×10^{-6}	**3.46×10^{-17}**	1.92×10^{-15}
	Std.dev.	6.56×10^{-4}	7.97×10^{-5}	**9.27×10^{-18}**	1.23×10^{-14}
f4	Min	1.83	3.52	**1.13×10^{-9}**	3.01×10^{-4}
	Max	5.46	7.88	**1.65×10^{-9}**	1.08×10^{-5}
	Mean	3.29	5.23	**1.60×10^{-9}**	1.78×10^{-6}
	Median	3.21	4.99	**1.32×10^{-9}**	1.14×10^{-6}
	Std.dev.	0.77	1.07	**2.20×10^{-10}**	2.32×10^{-6}
f5	Min	0.03	2.23×10^{-2}	21.27	**3.05×10^{-8}**
	Max	204.32	80.93	14.34	**4.95×10^{-3}**
	Mean	87.28	75.39	14.22	**4.51×10^{-4}**
	Median	76.88	69.21	14.24	**3.89×10^{-5}**
	Std.dev.	76.06	81.86	0.27	**1.05×10^{-3}**
f6	Min	**0**	**0**	**0**	**0**
	Max	**0**	4	**0**	**0**
	Mean	**0**	0.13	**0**	**0**
	Median	**0**	**0**	**0**	**0**
	Std.dev.	**0**	0.73	**0**	**0**
f7	Min	5.20×10^{-3}	2.40×10^{-2}	7.37×10^{-3}	**2.22×10^{-3}**
	Max	1.70×10^{-2}	5.22×10^{-2}	**1.19×10^{-2}**	7.42×10^{-2}
	Mean	1.07×10^{-2}	3.84×10^{-2}	1.14×10^{-2}	**5.19×10^{-3}**
	Median	1.00×10^{-2}	3.67×10^{-2}	1.11×10^{-2}	**5.43×10^{-3}**
	Std.dev.	2.90×10^{-3}	6.60×10^{-3}	2.14×10^{-3}	**1.30×10^{-3}**

Note: the bold values show the best results.

Table 6
The results of comparing three algorithms (GATC, GAMC and GSA) to TGA for the minimization of test functions in Table 2 with 10000 iterations.

		GATC	GAMC	GSA	TGA
f8	Min	25.86	58.70	7.96	**0**
	Max	85.56	202.97	21.88	**3.97**
	Mean	52.36	116.87	12.13	**0.63**
	Median	50.74	111.43	11.94	**6.95×10^{-8}**
	Std.dev.	15.03	36.15	2.61	**1.02**
f9	Min	6.84	9.68	1.80×10^{-9}	**7.99×10^{-15}**
	Max	13.95	18.04	2.47×10^{-9}	**1.86×10^{-14}**
	Mean	11.01	14.67	2.21×10^{-9}	**1.45×10^{-14}**
	Median	10.92	15.50	2.21×10^{-9}	**1.51×10^{-14}**
	Std.dev.	1.46	2.37	2.31×10^{-10}	**2.10×10^{-15}**
f10	Min	**0**	**0**	**0**	**0**
	Max	5.60	66.21	**0**	1.48×10^{-2}
	Mean	0.88	3.39	**0**	1.64×10^{-3}
	Median	0.39	0.10	**0**	6.37×10^{-9}
	Std.dev.	1.31	12.07	**0**	3.95×10^{-3}

Note: the bold values show the best result.

Please cite this article in press as: Mahmoodabadi MJ, et al., TGA: Team game algorithm, Future Computing and Informatics Journal (2018), https://doi.org/10.1016/j.fcij.2018.03.002

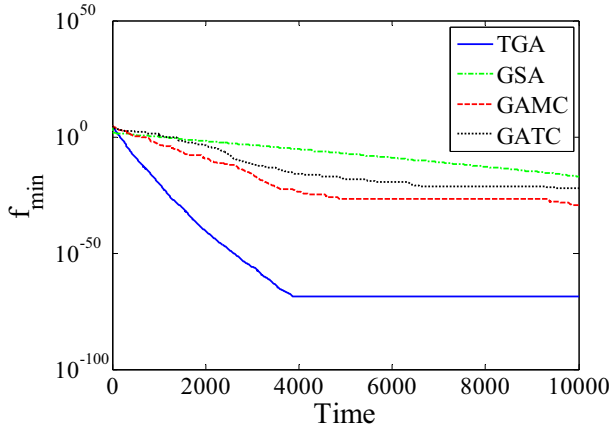

Fig. 5. The comparison of TGA, GSA, GAMC and GATC for the minimization of f1 with 10000 iterations (time) and 30 dimensions.

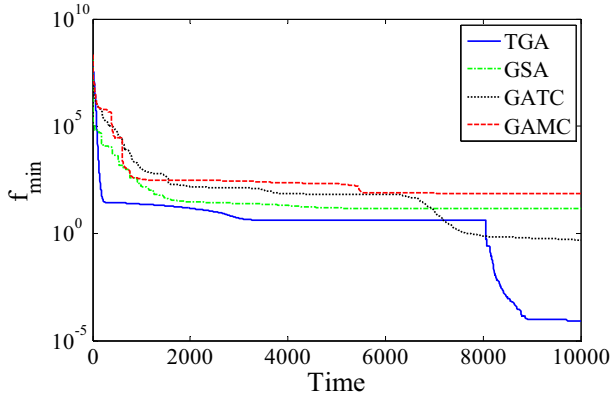

Fig. 6. The comparison of TGA, GSA, GAMC and GATC for the minimization of f5 with 10000 iterations (time) and 30 dimensions.

notable until about 4000 iterations which is the result of passing and mistake operators. Afterward, the substitution operator strikes out and brings out the algorithm from local minima even until about 8000 iterations and will let the other two operators play their roles again. This can be a verification to name TGA not only a speedy but also a precious algorithm.

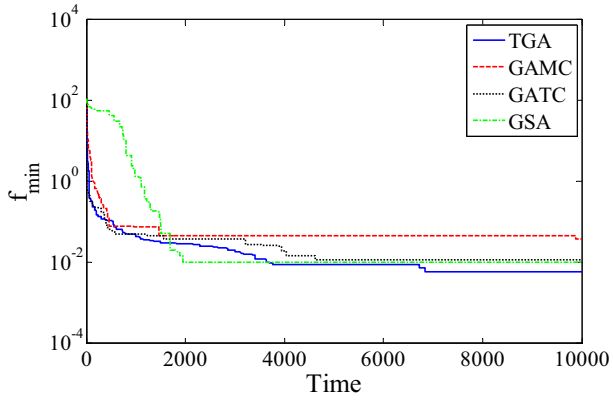

Fig. 7. The comparison of TGA, GSA, GAMC and GATC for the minimization of f7 with 10000 iterations (time) and 30 dimensions.

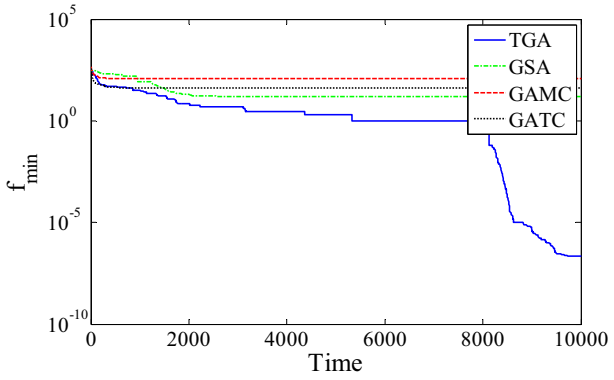

Fig. 8. The comparison of TGA, GSA, GAMC and GATC for minimization of f8 with 10000 iterations (time) and 30 dimensions.

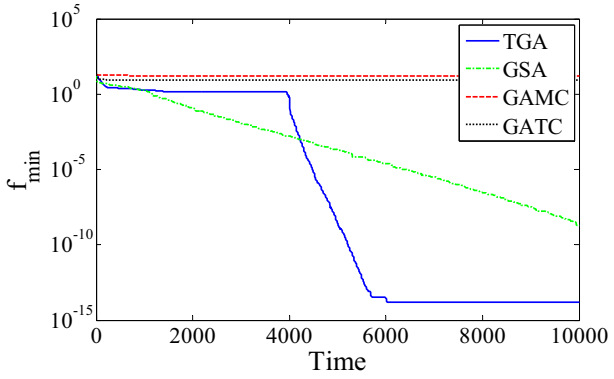

Fig. 9. The comparison of TGA, GSA, GAMC and GATC for the minimization of f9 with 10000 iterations (time) and 30 dimensions.

Table 7
Shifted unimodal test functions.

Function name	Formulation	Domain
g1: Sphere	$\sum_{i=1}^{n}(x_i-1)^2$	$[-100,100]^n$
g2: Schwefel 2.22	$\sum_{i=1}^{n}\lvert x_i-1\rvert+\prod_{i=1}^{n}\lvert x_i-1\rvert$	$[-10,10]^n$
g3: Quadric	$\sum_{i=1}^{n}(\sum_{j=1}^{i}(x_j-1))^2$	$[-100,100]^n$
g4: Schwefel 2.21	$\max_i\{\lvert x_i-1\rvert, 1\le i\le n\}$	$[-100,100]^n$
g5: Rosenbrock	$\sum_{i=1}^{n-1}[100((x_{i+1}-1)-(x_i-1)^2)^2+((x_i-1)-1)^2]$	$[-30,30]^n$
g6: Step	$\sum_{i=1}^{n}([(x_i-1)+0.5])^2$	$[-100,100]^n$
g7: Quadric noise	$\sum_{i=1}^{n}i(x_i-1)^4+random[0,1)$	$[-1.28,1.28]^n$

Notable points for these figures are:

- The convergence speed of the proposed method, due to a well-done operator named passing.
- Giving fantastic results in the Rosenbrock test function, due to logical passing and mistake operators.
- The effect of substitution operator when the other two operators couldn't affect noticeably anymore or the algorithm glitches in local minima for multimodal test functions.

Please cite this article in press as: Mahmoodabadi MJ, et al., TGA: Team game algorithm, Future Computing and Informatics Journal (2018), https://doi.org/10.1016/j.fcij.2018.03.002

Table 8
Shifted multimodal test functions.

Function name	Formulation	Domain
g8: Rastrigin	$\sum_{i=1}^{n}[(x_i-1)^2-10\cos(2\pi(x_i-1))+10]$	$[-5.12,5.12]^n$
g9: Ackley	$-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i-1)^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi(x_i-1))\right)+20+e$	$[-32,32]^n$
g10: Griewank	$\frac{1}{4000}\sum_{i=1}^{n}(x_i-1)^2-\prod_{i=1}^{n}\cos\left(\frac{(x_i-1)}{\sqrt{i}}\right)+1$	$[-600,600]^n$

As observed on the above tables, the results for TGA verify its stability for all test functions. As an example, although GSA gave better results in the unshifted Griewank function, it couldn't be superior in the specified shifted function. Tables 7–10 represent shifted unimodal test functions, shifted multimodal test functions, the results of comparing three algorithms (GATC, GAMC and GSA) to TGA for the minimization of the mentioned shifted functions in Table 7 with 10,000 iterations, and the results of comparing three algorithms (GATC, GAMC and GSA) to TGA for the minimization of the mentioned shifted functions in Table 8 with 10,000 iterations, correspondingly.

Table 9
The results of comparing three algorithms (GATC, GAMC and GSA) to TGA for the minimization of the mentioned shifted functions in Table 7 with 10000 iterations.

		GATC	GAMC	GSA	TGA
g1	Min	8.13×10^{-26}	7.86×10^{-30}	4.60×10^{-18}	**0**
	Max	1.76×10^{-20}	7.34×10^{-27}	8.90×10^{-18}	**0**
	Mean	8.42×10^{-22}	1.07×10^{-27}	6.87×10^{-18}	**0**
	Median	6.61×10^{-23}	4.37×10^{-28}	7.00×10^{-18}	**0**
	Std.dev.	3.22×10^{-21}	1.58×10^{-27}	1.14×10^{-18}	**0**
g2	Min	5.00×10^{-14}	1.47×10^{-14}	9.94×10^{-9}	**0**
	Max	4.47×10^{-9}	1.12×10^{-12}	1.74×10^{-8}	**0**
	Mean	2.19×10^{-10}	3.04×10^{-13}	1.29×10^{-8}	**0**
	Median	2.39×10^{-11}	1.46×10^{-13}	1.28×10^{-8}	**0**
	Std.dev.	8.09×10^{-10}	3.34×10^{-13}	2.16×10^{-9}	**0**
g3	Min	5.37×10^{-8}	7.56×10^{-10}	**2.52×10^{-17}**	2.16×10^{-16}
	Max	1.70×10^{-3}	1.36×10^{-4}	**5.95×10^{-17}**	2.14×10^{-14}
	Mean	2.76×10^{-4}	9.47×10^{-6}	**3.59×10^{-17}**	3.00×10^{-15}
	Median	5.59×10^{-5}	1.35×10^{-6}	**3.56×10^{-17}**	1.38×10^{-15}
	Std.dev.	5.08×10^{-4}	2.59×10^{-5}	**7.33×10^{-17}**	4.34×10^{-15}
g4	Min	20.14	30.20	**1.25×10^{-9}**	1.04×10^{-7}
	Max	46.68	63.77	**1.94×10^{-9}**	2.13×10^{-5}
	Mean	37.76	47.92	**1.56×10^{-9}**	2.69×10^{-6}
	Median	31.41	48.89	**1.55×10^{-9}**	1.41×10^{-6}
	Std.dev.	6.41	7.45	**1.95×10^{-9}**	4.39×10^{-6}
g5	Min	0.03	2.23×10^{-2}	21.27	**3.05×10^{-3}**
	Max	204.32	80.93	14.34	**4.95×10^{-3}**
	Mean	87.28	75.39	14.22	**4.51×10^{-4}**
	Median	76.88	69.21	14.24	**3.89×10^{-5}**
	Std.dev.	76.06	81.86	0.27	**1.05×10^{-3}**
g6	Min	**0**	**0**	**0**	**0**
	Max	9	**0**	**0**	**0**
	Mean	0.33	**0**	**0**	**0**
	Median	**0**	**0**	**0**	**0**
	Std.dev.	1.64	**0**	**0**	**0**
g7	Min	7.20×10^{-3}	1.52×10^{-2}	4.00×10^{-3}	**3.02×10^{-3}**
	Max	2.04×10^{-2}	5.77×10^{-2}	2.10×10^{-2}	**9.00×10^{-2}**
	Mean	1.19×10^{-2}	3.88×10^{-2}	1.48×10^{-2}	**6.14×10^{-3}**
	Median	1.16×10^{-2}	4.12×10^{-2}	1.56×10^{-2}	**5.83×10^{-3}**
	Std.dev.	3.00×10^{-3}	9.00×10^{-3}	4.10×10^{-3}	**1.62×10^{-3}**

Note: the bold values show the best results.

Table 10
The results of comparing three algorithms (GATC, GAMC and GSA) to TGA for the minimization of the mentioned shifted functions in Table 8 with 10000 iterations.

		GATC	GAMC	GSA	TGA
g8	Min	29.84	53.72	5.97	**0**
	Max	84.57	220.88	18.90	**1.98**
	Mean	49.94	117.33	12.80	**0.29**
	Median	51.24	110.44	11.94	**4.90×10^{-8}**
	Std.dev.	13.28	33.69	3.40	**0.59**
g9	Min	6.18	9.15	1.66×10^{-9}	**7.99×10^{-15}**
	Max	12.63	18.53	2.64×10^{-9}	**2.22×10^{-14}**
	Mean	10.47	14.72	2.10×10^{-9}	**1.48×10^{-14}**
	Median	10.63	14.99	2.09×10^{-9}	**1.51×10^{-14}**
	Std.dev.	1.56	2.54	2.59×10^{-9}	**3.08×10^{-15}**
g10	Min	**0**	7.40×10^{-3}	**0**	**0**
	Max	5.01	16.97	2.22×10^{-2}	**7.39×10^{-3}**
	Mean	0.55	0.86	1.60×10^{-3}	**9.88×10^{-4}**
	Median	0.21	0.13	**0**	3.57×10^{-7}
	Std.dev.	1.08	3.07	4.60×10^{-3}	**2.55×10^{-3}**

Note: bold values show the best results.

In Tables 7 and 8, test functions mentioned in Tables 1 and 2 have been shifted one unit in the direction of x axis. Therefore, all the shifted test functions have a minimum amount of 0 at $x = 1$, except for f5 which has a minimum amount of 0 at $x = 2$.

5. Conclusions

In this article, a new optimization method was introduced which unlike other algorithms, it is not stemmed from the natural phenomena but originated from the sport phenomena. According to this algorithm, team games consist of several skills. Sharp passing, learning from the mistakes and beside them, a well-timed substitution are some of these skills to achieve the common purpose of the team. Simulating vital rules of a team in order to minimize the optimization problems was the purpose of this article. Eventually, the proposed optimization algorithm was tested by several standard criterion functions and also was compared with three notable algorithms to declare the convergence speed and precision of finding the global optimum point in both shifted and unshifted test functions.

Please cite this article in press as: Mahmoodabadi MJ, et al., TGA: Team game algorithm, Future Computing and Informatics Journal (2018), https://doi.org/10.1016/j.fcij.2018.03.002

References

[1] Atashpaz-Gargari E, Lucas C. Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In: Proceedings of the IEEE Congress on Evolutionary computation; 2007. p. 4661−7.
[2] Chang WD. A multi-crossover genetic approach to multivariable PID controllers tuning. Expert Syst Appl 2007;33:620−6.
[3] Chu SC, Tsai PW, Pan JS. Cat swarm optimization. In: PRICAI 2006: trends in artificial intelligence. Springer; 2006. p. 854−8.
[4] Dorigo M, Maniezzo V, Colorni A. Ant system: optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics, Part B. Cybernetics 1996;26:29−41.
[5] Eberhart RC, Kennedy J. A new optimizer using particle swarm theory. In: Proceedings of the sixth international symposium on micro machine and human science. New York, NY; 1995. p. 39−43.
[6] Gandomi AH, Alavi AH. Krill herd: a new bio-inspired optimization algorithm. Commun Nonlinear Sci Numer Simulat 2012;17:4831−45.
[7] Geem ZW, Kim JH, Loganathan G. A new heuristic optimization algorithm: harmony search. Simulation 2001;76:60−8.
[8] Goldberg DE, Holland JH. Genetic algorithms and machine learning. Mach Learn 1988;3:95−9.
[9] Greiner R. PALO: a probabilistic hill-climbing algorithm. Artif Intell 1996;84:177−208.
[10] Holland JH. Genetic algorithms. Sci Am 1992;267:66−72.
[11] Hunt JE, Cooke DE. Learning using an artificial immune system. J Netw Comput Appl 1996;19:189−212.
[12] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Global Optim 2007;39:459−71.
[13] Katzenbach JR, Smith DK. The wisdom of teams: creating the high-performance organization. Harvard Business Press; 1993.
[14] Kirkpatrick S. Optimization by simulated annealing: quantitative studies. J Stat Phys 1984;34:975−86.
[15] Krishnanand K, Ghose D. Glowworm swarm optimisation: a new method for optimising multi-modal functions. Int J Comput Intell Stud 2009;1:93−119.
[16] Li XL, Qian JX. Studies on artificial fish swarm optimization algorithm based on decomposition and coordination techniques. J Circuits Syst 2003;1:1−6.
[17] Mirjalili S, Hashim SM. BMOA: binary magnetic optimization algorithm. In: Proceedings of the conference on machine learning and computing. Singapore; 2011. p. 201−6.
[18] Oftadeh R, Mahjoob M, Shariatpanahi M. A novel meta-heuristic optimization algorithm inspired by group hunting of animals: Hunting search. Comput Math Appl 2010;60:2087−98.
[19] Pan WT. A new fruit fly optimization algorithm: taking the financial distress model as an example. Knowl Base Syst 2012;26:69−74.
[20] Passino KM. Bacterial foraging optimization, innovations and developments of swarm intelligence applications. 2012. p. 219.
[21] Rajakumar B. The Lion's algorithm: a new nature-inspired search algorithm. Proced Technol 2012;6:126−35.
[22] Rashedi E, Nezamabadi-Pour H, Saryazdi S. GSA: a gravitational search algorithm. Inf Sci 2009;179:2232−48.
[23] Rechenberg I. Case studies in evolutionary experimentation and computation. Comput Meth Appl Mech Eng 2000;186:125−40.
[24] Rechenberg I. Evolution strategy: nature's way of optimization. In: Optimization: Methods and applications, possibilities and limitations. Springer; 1989. p. 106−26.
[25] Storn R, Price K. Differential evolution -a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 1997;11:341−59.
[26] Tayarani N, Akbarzadeh-T M. Magnetic optimization algorithms a new synthesis. In: Proceedings of the congress on evolutionary computation; 2008. p. 2659−64.
[27] Wang B, Jin X, Cheng B. Lion pride optimizer: an optimization algorithm inspired by lion pride behavior. Sci China Inf Sci 2012;55:2369−89.
[28] Wang G, Guo L, Gandomi AH, Cao L, Alavi AH, Duan H, et al. Lévy-flight krill herd algorithm, Mathematical Problems in Engineering. 2013.
[29] Xing B, Gao WJ. Fruit fly optimization algorithm. In: Innovative computational intelligence: a rough guide to 134 clever algorithms. Springer; 2014. p. 167−70.
[30] Yang XS. Firefly algorithm, Levy flights and global optimization. In: Research and development in intelligent systems XXVI. Springer; 2010. p. 209−18.
[31] Yang XS, Deb S. Cuckoo search via lévy flights. In: Procidingd of the congress on nature & biologically inspired computing; 2009. p. 210−4.
[32] Yang XS, Gandomi AH. Bat algorithm: a novel approach for global engineering optimization. Eng Comput 2012;29:464−83.

Please cite this article in press as: Mahmoodabadi MJ, et al., TGA: Team game algorithm, Future Computing and Informatics Journal (2018), https://doi.org/10.1016/j.fcij.2018.03.002