Engineering Applications of Artificial Intelligence 86 (2019) 165–181

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Poor and rich optimization algorithm: A new human-based and multi populations algorithm ☆

Seyyed Hamid Samareh Moosavi, Vahid Khatibi Bardsiri *

Department of Computer Engineering, Bardsir Branch' Islamic Azad University, Bardsir, Iran

ARTICLE INFO

Keywords:
Human based algorithm
Poor and rich optimization algorithm
Benchmark function
Engineering problems

ABSTRACT

This paper presents a new optimization algorithm called poor and rich optimization (PRO). This algorithm is inspired by the efforts of the two groups of the poor and the rich to achieve wealth and improve their economic situation. The rich always try to increase their class gap with the poor by gaining wealth from different ways. The rich are always trying to increase their class gap with the poor by acquiring wealth from different ways. On the other hand, the poor try to gain wealth and reduce their class gap with the rich. On the other hand, the poor try to gain wealth and reduce their class gap by modeling the rich. This struggle is always going on and should be mention that the poor may get rich and vice versa. The proposed algorithm is evaluated using 33 test functions and the simulation results are compared with a number of new and well-known optimization algorithms. The evaluation domain includes uni-modal, multi-modal, fixed dimension, hybrid and large scale functions. In addition, for more precise evaluation, Tension/compression spring design, pressure vessel design, Gear drain design, and three-bar truss design problems are solved by PRO algorithm. PRO algorithm has had better performance in these four problems by finding optimal values of parameters as compared to other algorithms. Finally, PRO algorithm was used to estimate software effort by UCP for more accurate evaluation. The obtained results confirmed the superiority of PRO in exploration, exploitation and convergence aspects, compared to other algorithms.

1. Introduction

In recent decades optimization has been one of the most important research areas. In fact, the optimization involves processes for finding the best answer for a particular problem. Principles and mechanisms of nature can influence the design of computational systems to solve complex problems. A large number of optimization algorithms are inspired by biological evolution. These algorithms are extensively used to solve engineering and scientific problems. Among common features, random operators have widely employed in optimization algorithms. The genetic algorithm (GA) (Holland, 1992; Holland and Reitman, 1977) is one of the most favorite optimization algorithms which include different types of randomness. Selection, reproduction, mutation and other processes are randomly performed in this algorithm, which significantly helps to prevent local optimum. In fact, random behavior is the main feature of optimization algorithms, which leads to non-deterministic results.

In some of bio-inspired algorithms, the collective behavior of animals or insects has been considered as the main idea behind the optimization process. This type of algorithms is located in the category of swarm intelligence algorithms. This category includes algorithms such as particle swarm optimization (PSO) (Kennedy and Eberhart, 1995), bat algorithm (BA) (Yang, 2010b), firefly algorithm (FA) (Yang, 2010a), social spider algorithm (SSO) (Cuevas et al., 2015), ant colony optimization (ACO) (Dorigo et al., 2006) and many others. Principals of nature considered in these algorithms have extended their optimization capability and have attracted the attention of researchers to generate novel swarm intelligence algorithms. A number of optimization algorithms are shown in Table 1.

On the other hands, the human behavior has been the main idea of some optimization algorithms called human-inspired algorithms which include tabu search (TS) (Glover, 1989, 1990), imperialist competitive algorithm (ICA) (Atashpaz-Gargari and Lucas, 2007), colliding bodies optimization (CBO) (Kaveh, 2014; Kaveh and Mahdavi, 2014), mine blast algorithm (MBA) (Sadollah et al., 2013a), seeker optimization algorithm (SOA) (Dai et al., 2007), group counseling optimization (GCO) algorithm (Eita and Fahmy, 2014, 2010) and Harmony Search (HS) (Geem et al., 2001).

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.engappai.2019.08.025.

* Corresponding author.
E-mail addresses: hamid.moosavi17@gmail.com (S.H. Samareh Moosavi), kvahid2@live.utm.my (V.K. Bardsiri).

https://doi.org/10.1016/j.engappai.2019.08.025
Received 16 November 2018; Received in revised form 20 August 2019; Accepted 30 August 2019
Available online xxxx
0952-1976/

Table 1
Optimization algorithm.

Algorithm	Year
Artificial Fish-Swarm Algorithm (AFSA) (Li, 2003)	2003
ABC (Artificial Bees Colony) (Karaboga and Basturk, 2007)	2006
CS (Cuckoo Search) (Yang and Deb, 2009)	2009
DPO (Dolphin Partner Optimization) (Shiqin et al., 2009)	2009
HS (Hunting Search) (Oftadeh et al., 2010)	2010
TLBO (Teaching–Learning-Based Optimization) (Rao et al., 2011)	2011
BMO (Bird Mating Optimizer) (Askarzadeh and Rezazadeh, 2012)	2012
KH (Krill Herd) (Gandomi and Alavi, 2012)	2012
DE (Dolphin Echolocation) (Kaveh and Farhoudi, 2013)	2013
GWO (Gray Wolf Optimization) (Mirjalili et al., 2014)	2014
ALO (Ant Lion Optimizer) (Mirjalili, 2015a)	2015
DA (Dragonfly Algorithm) (Mirjalili, 2015b)	2015
MFO (Moth–Flame Optimization) (Mirjalili, 2015c)	2015
MVO (Multi-Verse Optimizer) (Mirjalili et al., 2015)	2015
WOA (Whale Optimization Algorithm) (Mirjalili and Lewis, 2016)	2016
SCA (Sine Cosine Algorithm) (Mirjalili, 2016)	2016
MOGWO (Multi-Objective Grey Wolf Optimizer) (Mirjalilia et al., 2016)	2016
LOA (Lion Optimization Algorithm) (Yazdani and Jolai, 2016)	2016
WEO (Water Evaporation Optimization) (Kaveh and Bakhshpoori, 2016)	2016
SBO (Satin Bowerbird Optimizer) (Samareh Moosavi and Khatibi Bardsiri, 2017)	2017
GOA (Grasshopper Optimization Algorithm) (Saremi et al., 2017)	2017
RE-GWO (Random Walk Grey Wolf Optimizer) (Gupta and Deep, 2018)	2018
SELO (Socio Evolution & Learning Optimization Algorithm) (Kumar et al., 2018)	2018
HHO (Harris Hawks Optimizer) (Heidari et al., 2019)	2019
SFO (Sail Fish Optimizer) (Shadravan et al., 2019)	2019

Since existing optimization algorithms are unable to generate optimal results in any situation, new algorithms are proposed for better responses to complex optimization problems. In recent years extended versions of genetic algorithm have been introduced to solve special problems. For instance, the problem of less-than-container loading (Jamrus and Chien, 2016), open-shop scheduling (Hosseinabadi et al., 2018) and job-shop scheduling (Salido et al., 2016) has been solved by extended versions of genetic algorithm.

In comparison to bio-inspired algorithms, few human-inspired algorithms have recently been proposed. This paper attempts to propose a novel optimization algorithm based on human behavior. The proposed algorithm called PRO which is inspired by the rich and the poor efforts to improve their economical state and to increase their class gap (for the rich) and to reduce the class gap (for the poor). Although many optimization algorithms have been proposed in recent years, however there are few multi-population algorithms. The multi-population algorithm proposed in this paper is distinguished from similar algorithms by its optimization mechanism. Indeed, the mechanism of PRO optimization in each population is different and this feature is the main reason for optimizing the proposed algorithm that has not yet been experienced. This paper includes four sections: Section 2 introduces different steps of PRO. Section 3 presents the experimental results. Section 4 presents the discussion. Finally, conclusion and future work are elaborated in Section 5.

2. Poor and rich optimization algorithm

PRO is designed based on a real social phenomenon which can be considered as a solution for complicated optimization problems. This section explains the optimization process of the PRO algorithm in five steps.

2.1. Background and inspiration

Wealth is one of the common concepts that are used in different subject fields. in particular, wealth is one of the economic concepts that according to the kind of attitude and implementation place different definitions have been defined. Wealth can be defined as determining the economic status of each individual whose quality and quantity are defined in the division of economic categories. There is probably no one in the world who has dreamed of becoming wealthy. There is a financial approach to every individual, and people are ambitious about their needs and desires for money. Although it is possible to inquire ways to get rich from any thoughtful man, sociologist or financial management expert, but using the experience and awareness of the approach of the richest people in the world seems to be most effective. Generally, people in a society can be categorized in two financial classes. The first category consists of rich individuals, whose wealth level is higher than average. The second group includes the poor that their wealth level is lower than average. Each member of these two groups is trying to improve their economic conditions in different ways, which have a great variety of diversity. But the common point that exists between all the members of the two groups is that individuals examine the behavior of each other and try to improve their position by influencing from them. More precisely, the poor population tries to improve their situation by targeting rich populations and rich populations try to increase their class gap by considering the poor population. In fact, the main idea of the proposed algorithm is summarized in two cases:

1. Each member of the poor group tries to improve their status and reduce their class gap by learning from the rich.
2. Each member of the rich group tries to increase the class gap with the poor by observing them and acquiring wealth.

The first population has been chosen randomly using uniform function in MATLAB Software. This kind of population was presented as a vector by using uniform function. The values of this vector have been specified between the upper bound and lower bound parameters using this function. The first population including the rich and poor was randomly designed as a uniform vector and this population will be arranged based on the objective function. The first part of this population including the best values will be considered for the rich group and the second part will be defined for the poor group.

2.2. Initial population

In the PRO algorithm, an initial population is randomly generated with uniform distribution between upper bound and lower bound parameters. Then, this initial population is evaluated and sorted in an ascending order based on the objective function's results. The main population in the PRO algorithm consists of two subpopulations. The first subpopulation is related to the rich while the second one is related to the poor. The size of these two subpopulations depends on the problem. The Eq. (1) shows the main population in the PRO algorithm:

$$POP_{Nmain} = POP_{Npoors} + POP_{Nriches} \tag{1}$$

where POP_{Nmain}, POP_{Npoors} and $POP_{Nriches}$ demonstrate the size of main population, poor population and rich population, respectively. Since the main population is sorted in an ascending order, its first part is related to the rich population and the second part is related to the poor population. In fact, in the PRO algorithm, all the rich population members have better positions than the poor population members. Fig. 1 shows these two populations.

The Eq. (2) is always correct in PRO:

$$\begin{aligned} &\text{cost}_1 < \text{cost}_2 < cost_3 < \cdots < cost_{\text{m}} < \text{cost}_{\text{m+1}} < \text{cost}_{\text{m+2}} \\ &< cost_{\text{m+3}} < \cdots < cost_{\text{n}} \end{aligned} \tag{2}$$

2.3. Updating the positions

The main population in the PRO algorithm consists of the poor and the rich subpopulations. In any iteration of the algorithm, the position of each population member must be changed according to a defined mechanism.

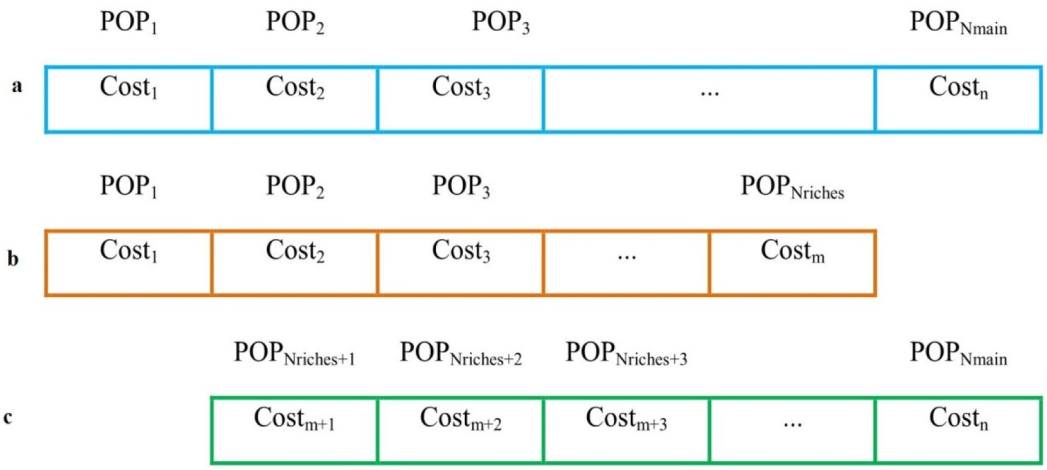

Fig. 1. The main (a), rich (b) and poor (c) populations.

2.3.1. The change in the position of each rich population member

The position of each member of the rich population changes according Eq. (3) in any iteration of the PRO algorithm:

$$\overrightarrow{X^{new}_{rich,i}} = \overrightarrow{X^{old}_{rich,i}} + r\left[\overrightarrow{X^{old}_{rich,i}} - \overrightarrow{X^{old}_{poor,best}}\right] \quad (3)$$

where $\overrightarrow{X^{new}_{rich,i}}$ is the new value of the ith position of the rich population, $\overrightarrow{X^{old}_{rich,i}}$ is the present value of the ith position of the rich population, r is the class gap parameter and $\overrightarrow{X^{old}_{poor,best}}$ is the present position of the best member of the poor population. The value of X is accounted as a vector of all variables. In fact, each member of the rich population increases its distance from every member of the poor population by getting richer. Since $\overrightarrow{X^{old}_{poor,best}}$ is the best member of the poor population, when a member of the rich population increases the distance from $\overrightarrow{X^{old}_{poor,best}}$, its distance from all members of the poor population is increased. In fact, the distance between the poor and the rich gets higher when the poor population gets poorer. Fig. 2 shows how a member of the rich population gets richer and the distance from the poor population gets higher.

Based on Fig. 2, $X^{old}_{rich,i}$ is the current position of ith member of the rich population, $X^{new}_{rich,i}$ is the new position of ith member of the rich population, $X^{old}_{poor,1}$ is the best member of the poor population. D1 is the distance between $X^{new}_{rich,i}$ and $X^{old}_{poor,1}$. By imposing the random coefficient of r, a distance will be created between $X^{new}_{rich,i}$ and $X^{old}_{poor,1}$ as the size of D1 and it will affect the positions of $X^{new}_{rich,i}$ and $X^{old}_{poor,1}$ as the size of D1+rD1. Since $X^{old}_{poor,1}$ is accounted as the best member of the poor population, so the distance of $X^{new}_{rich,i}$ from the other members of the poor population will be increased. In fact, $X^{old}_{poor,1}$ which was shown in Fig. 2 is the same as $X^{old}_{poor,best}$ in Eq. (3). Similarly, the value of D1 will be negative when the value of $X^{old}_{poor,1}$ is higher than the value of $X^{old}_{rich,i}$. This negative value improves the current position of the poor population. In the better possible situation, the value of $X^{old}_{poor,1}$ will be equal to the value of $X^{old}_{rich,i}$ as the value of D1 will be 0 and no changes will be occurred.

r is a random value between 0 and 1 which randomly determines the distance that each member of the rich population should keep from the poor population. Randomness of r causes an internal competition in the rich population. In other words, when two members of the rich population are close in position and $\overrightarrow{X^{old}_{poor,best}}$ is fixed, r determines the improvement of each member. It means that positions with higher costs may get more improvements than positions with lower costs by using higher r value.

2.3.2. The change in the position of each poor population member

The position of each poor population member changes according to Eq. (4) in each PRO algorithm iteration:

$$\overrightarrow{X^{new}_{poor,i}} = \overrightarrow{X^{old}_{poor,i}} + [r\,(Pattern) - \overrightarrow{X^{old}_{poor,i}}] \quad (4)$$

where $\overrightarrow{X^{new}_{poor,i}}$ is the new value of the ith position of the poor population, $\overrightarrow{X^{old}_{poor,i}}$ is the present value of the ith position of the poor population, r is the pattern improvement parameter (with a random value between 0 and 1) and the pattern of getting rich. The Eq. (5) gives the pattern value:

$$Pattern = \frac{\overrightarrow{X^{old}_{rich,\,best}} + \overrightarrow{X^{old}_{rich,\,mean}} + \overrightarrow{X^{old}_{rich,\,worst}}}{3} \quad (5)$$

where $\overrightarrow{X^{old}_{rich,best}}$ is the best member of positions in the rich population, $\overrightarrow{X^{old}_{rich,mean}}$ is the average position of the rich population members while $\overrightarrow{X^{old}_{rich,worst}}$ is the position of the worst member in the rich population. Since getting rich varies for each person (getting rich is a fuzzy parameter), the average position of three representatives of the rich is targeted in each iteration. Considering that fact that the pattern value is fixed in each iteration, the random parameter r determines the pattern improvement and consequently the improvement in $\overrightarrow{X^{old}_{poor,i}}$. In fact, when the r value is close to 0, more improvement is seen in the pattern; and since $\overrightarrow{X^{old}_{poor,i}}$ is greater than the pattern value, it will have a greater improvement and vice versa. Randomness of r causes an internal competition in the poor population. It means that the smaller value of r leads to more improvement in the pattern, and where the positions are close to each other, the r value causes changes in rank of these positions.

2.4. Mutation

There are factors in the world of economy that can improve or not improve the economic situation. for instance: a sharp fall or rise in the gold price in a short period of time, a sharp increase or decrease in the oil or petrochemical production prices, a sharp fall or rise in exchange rates, a sharp fall or rise in the stocks interest rates, a sharp fall or rise in the banking interests and etc. Each of these factors can cause a sudden change in some people's situation in a society. Since the prediction of these factors is very difficult and in some cases it is impossible to predict that this factor is used in this algorithm as a mutation. In this algorithm, a normal distribution with zero mean and variance of one and with a given probability for the mutation on the population of the poor and the rich is used separately.

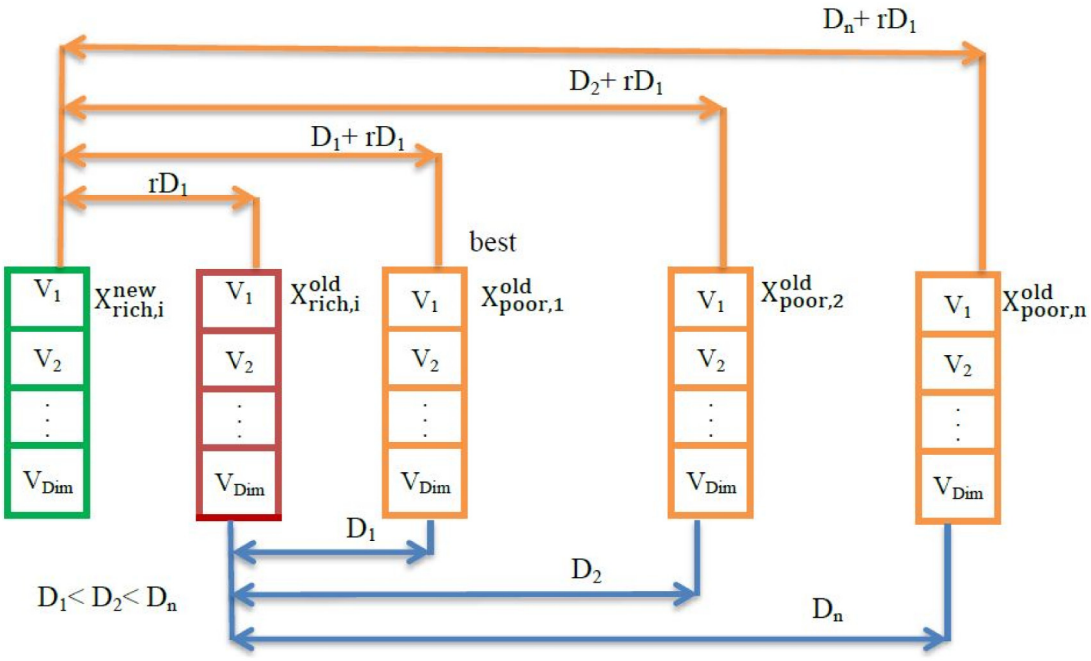

Fig. 2. The distance between poor and rich populations.

The Eqs. (6) and (7) determine the mutation for the rich and the poor populations, respectively.

if rand < Pmut

$$\overrightarrow{X^{new}_{rich,\,i}} = \overrightarrow{X^{new}_{rich,\,i}} + randn \tag{6}$$

end

if rand < Pmut

$$\overrightarrow{X^{new}_{poor,\,i}} = \overrightarrow{X^{new}_{poor,\,i}} + randn \tag{7}$$

end

where *rand* is a random value between 0 and 1, Pmut is the mutation probability, $X^{new}_{rich,i}$ is the new value from Eq. (3) before performing the mutation on the same position, $X^{new}_{poor,i}$ is the new value from Eq. (4) before performing the mutation and randn is the value resulted from the normal distribution with the average of 0 and variance of 1. As it is obvious from Eqs. (6) and (7), the mutation is applied to new positions that may lead to improvement or not.

2.5. Making a new population after each iteration

There are four populations after any iteration of the PRO algorithm as follows:

1. The old population of the poor
2. The old population of the rich
3. The new population of the poor
4. The new population of the rich

These four populations are evaluated by the objective function and are combined at the end of each iteration. The composite population is based on the values obtained from the objective function in ascending order, and the two subpopulations of the rich and the poor are separated by the number defined prior to the beginning of this compound population. The reason for the combination of the poor and the rich at the end of any iteration is that a member of the poor population may have acquired enough wealth to replace one of the rich. On the contrary, a rich man may replace a member of the poor by losing his wealth. By the way, the best member is always the first one in the rich population. The PRO algorithm flowchart is shown in Fig. 3.

Table 2
Unimodal functions.

Function	Dim	Range	f_{min}
$F_1(X) = \sum_{i=1}^{n} X_i^2$	30	[−100,100]	0
$F_2(X) = \sum_{i=1}^{n} \lvert X_i \rvert + \prod_{i=1}^{n} \lvert X_i \rvert$	30	[−10,10]	0
$F_3(X) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} X_j\right)^2$	30	[−100,100]	0
$F_4(X) = \max\{\lvert X_i \rvert, 1 \le i \le n\}$	30	[−100,100]	0
$F_5(X) = \sum_{i=1}^{n-1} \left[100\left(X_{i+1} - X_i^2\right)^2 + \left(X_i - 1\right)^2\right]$	30	[−30,30]	0
$F_6(X) = \sum_{i=1}^{n} \left(\lfloor X_i + 0.5 \rfloor\right)^2$	30	[−100,100]	0
$F_7(X) = \sum_{i=1}^{n} iX_i^4 + random[0,1)$	30	[−1.28,1.28]	0
$F_8(X) = X_1^2 + 10^6 \sum_{i=2}^{n} X_i^2$	30	[−100,100]	0
$F_9(X) = \sum_{i}^{n} \lvert X_i \rvert^{i+1}$	30	[−100,100]	0
$F_{10}(X) = \sum_{i=1}^{n} X_i^2 + \left(\sum_{i=1}^{n} 0.5X_i\right)^2 + \left(\sum_{i=1}^{n} 0.5X_i\right)^4$	30	[−100,100]	0

3. Experimental results

The average (AVE), the standard deviation (STD), the statistical analysis and the convergence of the results of the algorithms on unimodal, multi-modal, fixed-dimension multimodal and hybrid test functions are compared in this section. Wilcoxon test rank sum is also used for the purpose of comparison (Derrac et al., 2011). This statistical test has a *p*-value that determines the significance level of differences among algorithms. The difference is considered significant if *p*-value is lower than 0.05. To evaluate the performance of the PRO algorithm, its results are compared to PSO (the most well-known algorithm among swarm-based algorithms), SCA and ABC algorithms. Also, for further investigation of the PRO algorithm efficiency, the results obtained from this algorithm are compared with the last 6 algorithms (from 2014 to 2017), which are fully illustrated in Section 3.2. In fact, PRO algorithm efficiency has been measured in two steps. The average of 30 times of running has been considered as the final result. The parameters of the algorithms employed in this work are calibrated by trial and error. For the maximum iteration parameter, three values of 250, 500 and 1000 were considered and it should mention that all algorithms have achieved the most optimal value of 1000 as the output. For the initial population parameter, values of 30 and 50 are considered, and

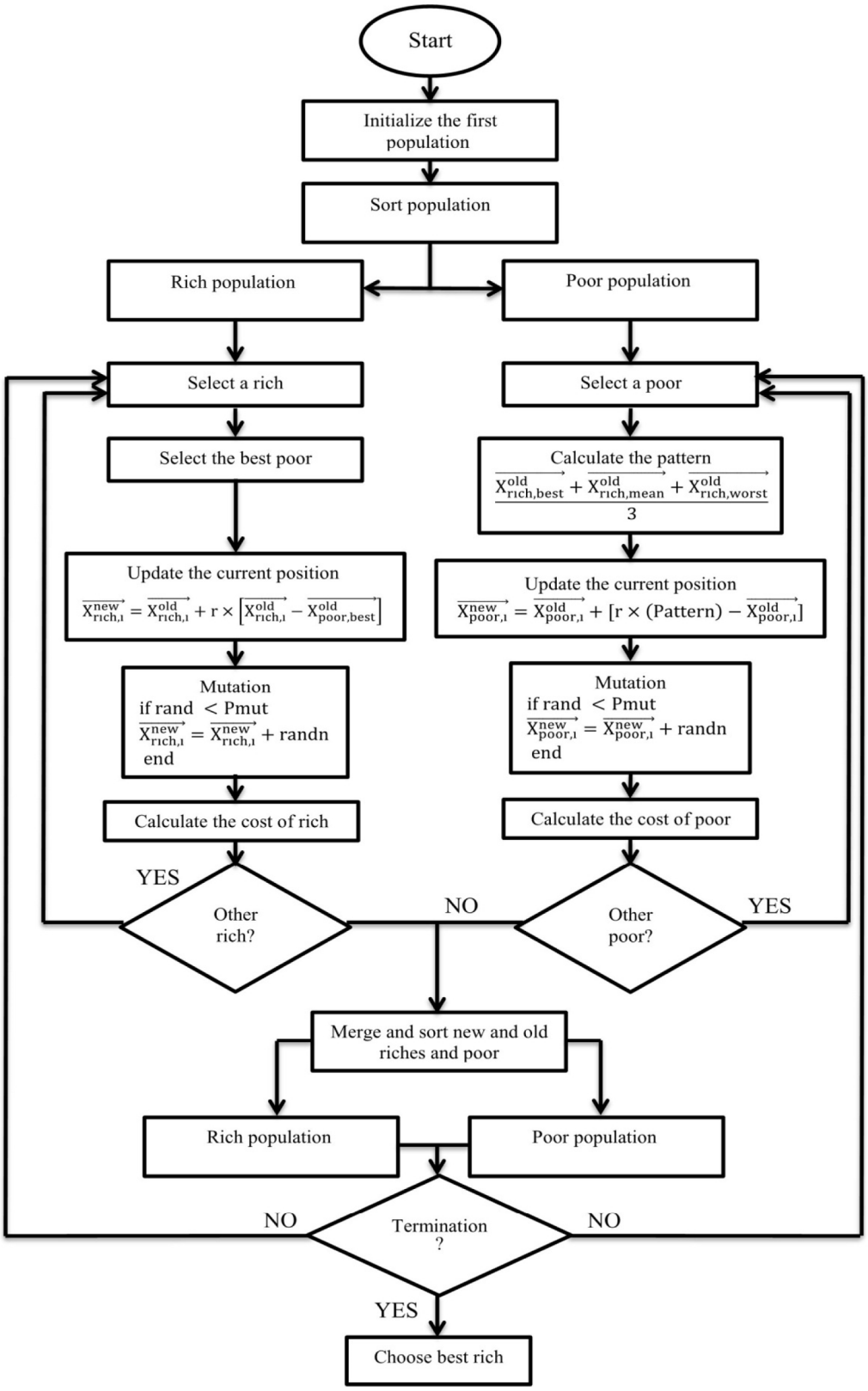

Fig. 3. The PRO algorithm.

the decision of 50 for all algorithms results in an optimal output. In the PRO algorithm, the mutation probability parameter [0.05, 0.09] is considered where the value of 0.06 was the most appropriate. The parameters C1 and C2 of the PSO algorithm are considered in Holland (1992) and Holland and Reitman (1977), which are an appropriate value for both parameters as well as a value of 1 for the W parameter is chosen. In the ABC algorithm, for the limit parameter the interval value [100,200] is selected, which eventually selection of 100 caused the optimal solution for this algorithm.

Table 3
The results of the algorithms on unimodal functions.

Functions		PRO	SCA	ABC	PSO
F1	AVE	**0**	3.76E−4	4.44	6.26E−14
	STD	**0**	5.51E−4	3.26	1.62E−13
	RANK	**1**	3	4	2
F2	AVE	**0**	4.13E−6	9.68E+1	2.07E−4
	STD	**0**	8.93E−6	22.60	3.06E−4
	RANK	**1**	2	4	3
F3	AVE	**0**	3.28E+3	5.93E+4	2.16E+2
	STD	**0**	2.88E+3	9.51E+3	54.74
	RANK	**1**	3	4	2
F4	AVE	**0**	19.58	55.84	2.81
	STD	**0**	10.80	4.68	0.67
	RANK	**1**	3	4	2
F5	AVE	**5.73E−14**	48.27	1.25E+6	92.07
	STD	**1.99E−13**	25.25	7.33E+5	93.07
	RANK	**1**	2	4	3
F6	AVE	**4.16E−20**	4.22	3.50	5.42E−18
	STD	**7.88E−20**	0.41	2.168	5.78E−18
	RANK	**1**	4	3	2
F7	AVE	**3.43E−5**	0.02	0.37	0.01
	STD	**2.36E−5**	0.01	0.11	3.1E−3
	RANK	**1**	3	4	2
F8	AVE	**0**	0.02	2.62E+6	9.69E−6
	STD	**0**	0.021	1.80E+6	1.30E−5
	RANK	1	3	4	2
F9	AVE	**0**	2.85E−6	1.42E+7	38.36
	STD	**0**	2.88E−6	9.60E+6	12.24
	RANK	1	2	4	3
F10	AVE	**0**	0.16	3.52E+4	1.02E+2
	STD	**0**	0.17	2.08E+4	46.01
	RANK	**1**	2	4	3

3.1. The unimodal, multi-modal, fixed-dimension and hybrid functions results

Unimodal functions possess a global optimum and no local optimum. These functions are proper to determine the performance of the proposed algorithm. Table 2 shows these functions.

Table 3 compares the PRO algorithm results with the results obtained from the other algorithms. It is observed that PRO can show the best results. It can find 0 (the best possible value) in F_1, F_2, F_3, F_4, F8, F9, F10 functions. All the P values are less than 0.05 and this confirms the superiority of PRO algorithm against the other algorithms. Fig. 4 shows the convergence trend of the algorithms on F_5, F_6 and F_7 functions. As it is clear from the figure, the proposed algorithm converges to the fewer number of function evaluation and obtains the best results compared to other algorithms. Since PRO algorithm is along with two kinds of population (poor and rich), so the recalls' number of the evaluation function within this algorithm is two times more than the other algorithms.

Multi-modal functions possess numerous local optimums. These functions are proper for the operation of exploration and avoiding local optimal algorithms. These functions are shown in Table 4.

Table 5 shows the results of PRO on 8 multi-modal test functions. As the table shows, the PRO algorithm has the best results in all functions compared to the other algorithms. PRO can also find the minimum value of 0 in F_{12} and F_{14} functions. In functions F17 and F18, the results obtained from the PRO algorithm have a significant difference with the results of the other algorithms. The results confirm that the difference between the PRO algorithm and the other algorithms is statistically significant because all the P values are less than 0.05. Fig. 5 shows the convergence trend of the algorithms on some multi-modal test functions. As the image shows, the PRO algorithm converges to the fewer number of function evaluation and obtains the best results compared to other algorithms. The high exploration level of the PRO algorithm enables this algorithm to avoid local optimums and move toward a global optimum.

Table 6 shows the fixed-dimension multimodal along with the exploration range and their optimum values.

Table 7 shows the results of the optimization algorithms on 10 fixed-dimension multimodal test functions. As the results show, the PRO algorithm is ranked 4th in F_{19} and F_{25} functions and 1st in the other functions. Although other algorithms also find the optimum answer in F_{21}, F_{22} and F_{23} functions, in F_{20}, F_{24}, F_{26}, F_{27} and F_{28} functions only PRO can find the best answer. The results of the statistical test show that all values are less than 0.05. Due to the obtained results, the algorithm is statistically superior to other algorithms. In F19 and F25 Since, PRO algorithm is along with an effective parameter, so this algorithm was not associated with a desirable performance within these two functions. In fact, the mutation operator did not affect these two functions. Fig. 6 shows the convergence trend of the algorithms on

Table 4
Multi-modal functions.

Function	Dim	Range	f_{min}
$F_{11}(X) = \sum_{i=1}^{n} -X_i \sin\left(\sqrt{\lvert X_i \rvert}\right)$	30	[−500,500]	−418*Dim
$F_{12}(X) = \sum_{i=1}^{n} \left[X_i^2 - 10\cos(2\pi X_i) + 10\right]$	30	[−5.12,5.12]	0
$F_{13}(X) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} X_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi X_i)\right) + 20 + e$	30	[−32,32]	0
$F_{14}(x) = \frac{1}{4000}\sum_{i=1}^{n} X_i^2 - \prod_{i=1}^{n}\cos\frac{X_i}{\sqrt{i}} + 1$	30	[−600,600]	0
$F_{15}(X) = \frac{\pi}{n}\left\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i-1)^2\left[1+10\sin 2(\pi y_{i+1})\right] + (y_n-1)^2\right\} + \sum_{i=1}^{n} u(X_i,10,100,4)$, $y_i = 1 + \frac{X_i+1}{4}$, $u(X_i,a,k,m) = \begin{cases} k(X_i-a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-X_i-a)^m & x_i < a \end{cases}$	30	[−50,50]	0
$F_{16}(x) = 0.1\left\{\sin 2(3\pi x_1) + \sum_{i=1}^{n}(x_i-1)^2\left[1+\sin 2(3\pi x_i+1)\right] + (x_n-1)^2\left[1+\sin 2(2\pi x_n)\right]\right\} + \sum_{i=1}^{n} u(X_i,5,100,4)$	30	[−50,50]	0
$F_{17}(X) = \sum_{i=1}^{n-1}\left(100\left(X_i^2 - X_{i+1}\right)^2 + (X_i-1)^2\right)$	30	[−100,100]	0
$F_{18}(X) = \sin 2(\pi w_1) + \sum_{i=1}^{n-1}(w_i-1)^2\left[1+10\sin 2(\pi w_i+1)\right] + (w_n-1)^2\left[1+\sin 2(2\pi w_n)\right]$, $w_i = 1 + \frac{x_i-1}{4}, \quad \forall i = 1,\ldots,n$	30	[−100,100]	0

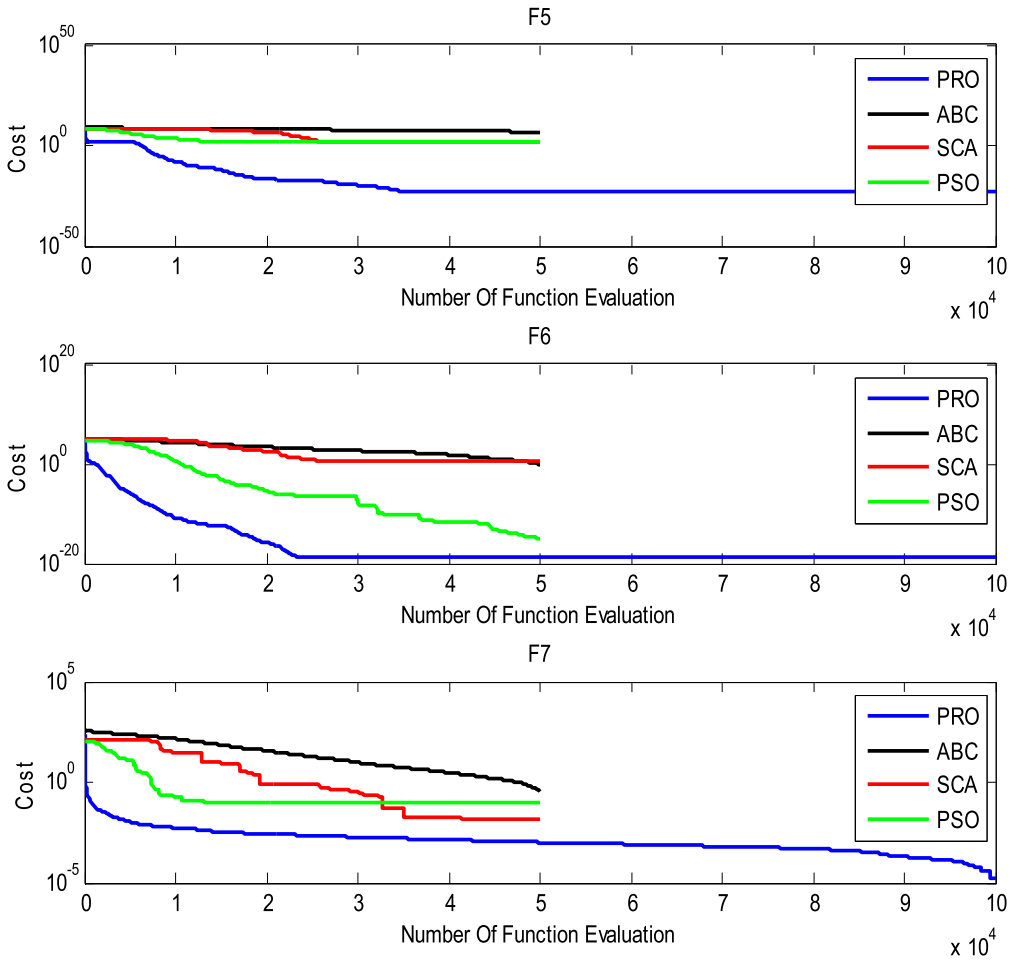

Fig. 4. Convergence trend of the algorithms on unimodal functions.

some fixed-dimension multimodal test functions. As the image shows, the convergence of PRO are less than those of the other algorithms.

In Table 8, Hybrid 2017 functions are shown. The results obtained from these functions related to the various algorithms are compared (As mentioned in Table 9). As it is clear from the results, the PRO algorithm has obtained the best value in terms of the mean and standard deviation for all hybrid functions. The PRO algorithm has reached the lowest possible value by gaining 0 within F29, F30, and F33 functions.

The PRO algorithm achieves 0.27 within F32 function, which (0.27 value) is half as the value obtained from the SCA algorithm. The results obtained from F31 function for the three algorithms (ABC, PSO and SCA) are not significantly different, but the value of the PRO algorithm is significantly different from these three algorithms.

In Fig. 7, the convergence diagram is shown for the two hybrid functions: F31 and F32. The PRO algorithm has found the minimum values for the variables of these two functions to the other algorithms.

3.2. PRO results versus recently algorithms

In this section, the results obtained from PRO are compared to those of reported in recent research studies. 33 test functions are employed to compare the results of PRO with six optimization algorithms. The initial population is 50 and 1000 for all algorithms and iteration, respectively. In the PRO algorithm, the effective parameter is the mutation probability, as the value of this parameter is considered 0.06. The results obtained for all algorithms are accounted as 30 runs in average. The initial settings for the parameters of these algorithms are considered (As mentioned in Section 3). Table 10 shows the results obtained from unimodal and multi-modal functions.

Table 5
The results from the exploration algorithms on multi-modal functions.

Functions		PRO	SCA	ABC	PSO
F11	AVE	**−9.06E+3**	−4.00E+3	−6.30E+3	−6.70E+3
	STD	944.85	**223.69**	978.86	757.79
	RANK	**1**	4	3	2
F12	AVE	**0**	8.28	2.24E+2	49.41
	STD	**0**	9.54	13.93	14.42
	RANK	**1**	2	4	3
F13	AVE	**8.88E−16**	0.28	3.01	5.22E−9
	STD	**1.00E−31**	0.63	0.68	8.00E−9
	RANK	**1**	3	4	2
F14	AVE	**0**	0.19	0.95	0.02
	STD	**0**	0.21	0.13	0.01
	RANK	**1**	3	4	2
F15	AVE	**4.48E−21**	0.61	3.15E+6	3.76E−19
	STD	**1.18E−20**	0.20	1.46E+6	6.82E−19
	RANK	**1**	3	4	2
F16	AVE	**2.77E−21**	2.78	4.41E+6	4.71E−14
	STD	**5.37E−21**	0.88	2.27E+6	1.04E−13
	RANK	**1**	3	4	2
F17	AVE	**2.74E−20**	39.12	7.53E+7	1.63E+2
	STD	**6.24E−20**	23.12	5.26E+7	26.93
	RANK	**1**	2	4	3
F18	AVE	**6.59E−21**	9.05	4.35E+3	1.86E+3
	STD	**2.50E−20**	8.30	3.20E+3	4.56E+3
	RANK	**1**	2	4	3

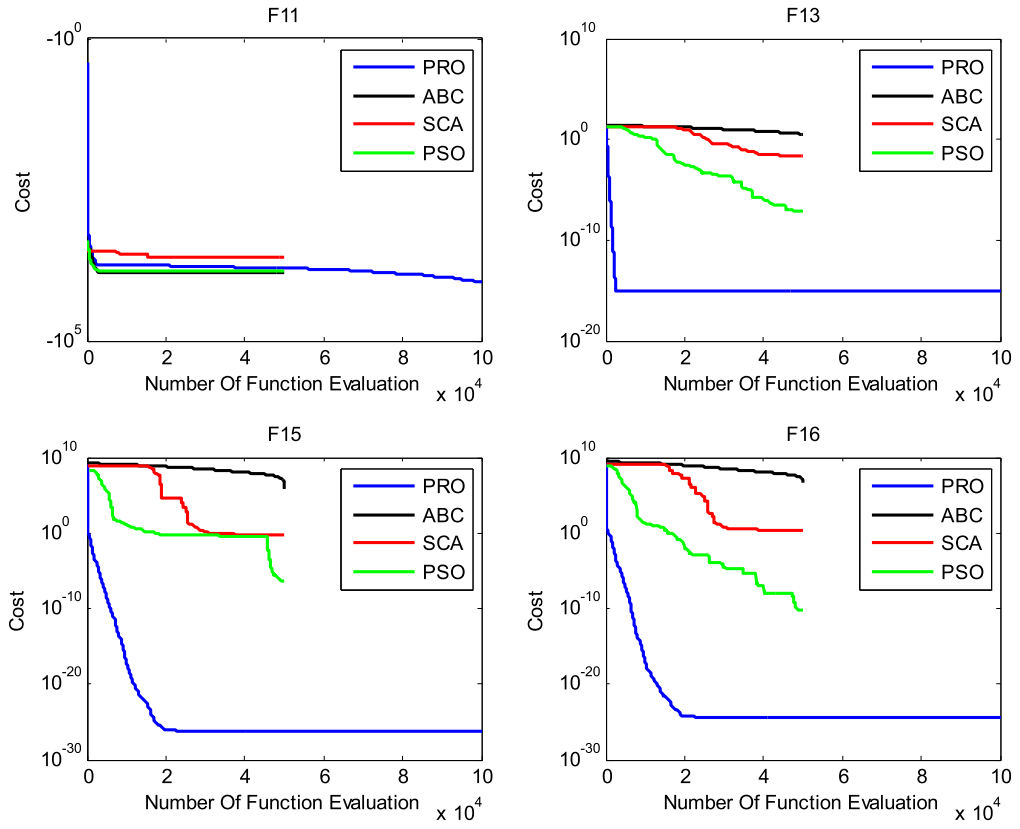

Fig. 5. The convergence trend of the algorithms on multi-modal functions.

Table 6
Fixed-dimension multimodal functions.

Function	Dim	Range	F_{nmi}
$F_{19}(x)=\left(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right)^{-1}$	2	[−65,65]	1
$F_{20}(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_i x_2)}{b_i^2+b_i x_3+x_4}\right]^2$	4	[−5,5]	0.00030
$F_{21}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	2	[−5,5]	−1.0316
$F_{22}(x)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1+6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos x_1+10$	2	[−5,5]	0.398
$F_{23}(x)=\left[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)\right]\times\left[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)\right]$	2	[−2,2]	3
$F_{24}(x)=-\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2\right)$	3	[1,3]	−3.86
$F_{25}(x)=-\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{6}a_{ij}(x_j-p_{ij})^2\right)$	6	[0,1]	−3.32
$F_{26}(x)=-\sum_{i=1}^{5}\left[(X-a_i)(X-a_i)^T+c_i\right]^{-1}$	4	[0,10]	−10.1532
$F_{27}(x)=-\sum_{i=1}^{7}\left[(X-a_i)(X-a_i)^T+c_i\right]^{-1}$	4	[0,10]	−10.4028
$F_{28}(x)=-\sum_{i=1}^{10}\left[(X-a_i)(X-a_i)^T+c_i\right]^{-1}$	4	[0,10]	−10.5363

Based on the results, the PRO algorithm obtained the best results than the other 6 algorithms within the unimodal functions (F1–F10). The PRO algorithm gains the value of 0 on the 7 functions (F1–F4 and F8–F10), which is the lowest (minimum) value for all these functions. The GWO and WOA algorithms have obtained the results on F1, F2, F8, F9 functions which are close to the PRO algorithm, but the average of these results was not equal to 0. On F8 function, the result of GWO algorithm was close to the results of PRO algorithm. The results obtained from the F5, F6, and F7 functions show that the difference between the results of the PRO algorithm and other algorithms is significant. In multimodal functions (F11–F18), the PRO algorithm has gained the best value for all functions other than F11. In F11 function, the WOA algorithm has gained the minimum value compared to the other algorithms. On the F12 and F14 functions, PRO and WOA algorithms gained a value of 0 which is the lowest (minimum) and common value for these functions. In Table 11, the results of PRO are investigated on fixed–dimension multimodal and hybrid functions.

The PRO algorithm has obtained the best results compared to the other comparable algorithms on 8 functions of 10 multimodal ones. In fact, the PRO algorithm failed to achieve an average value on F19 and F25 functions. On F21, F22, and F23 functions, all algorithms have reached the minimum value. The PRO algorithm achieved the minimum value to the other algorithms on F20 and F26 and F27 functions. The WOA algorithm was able to obtain the lowest value from

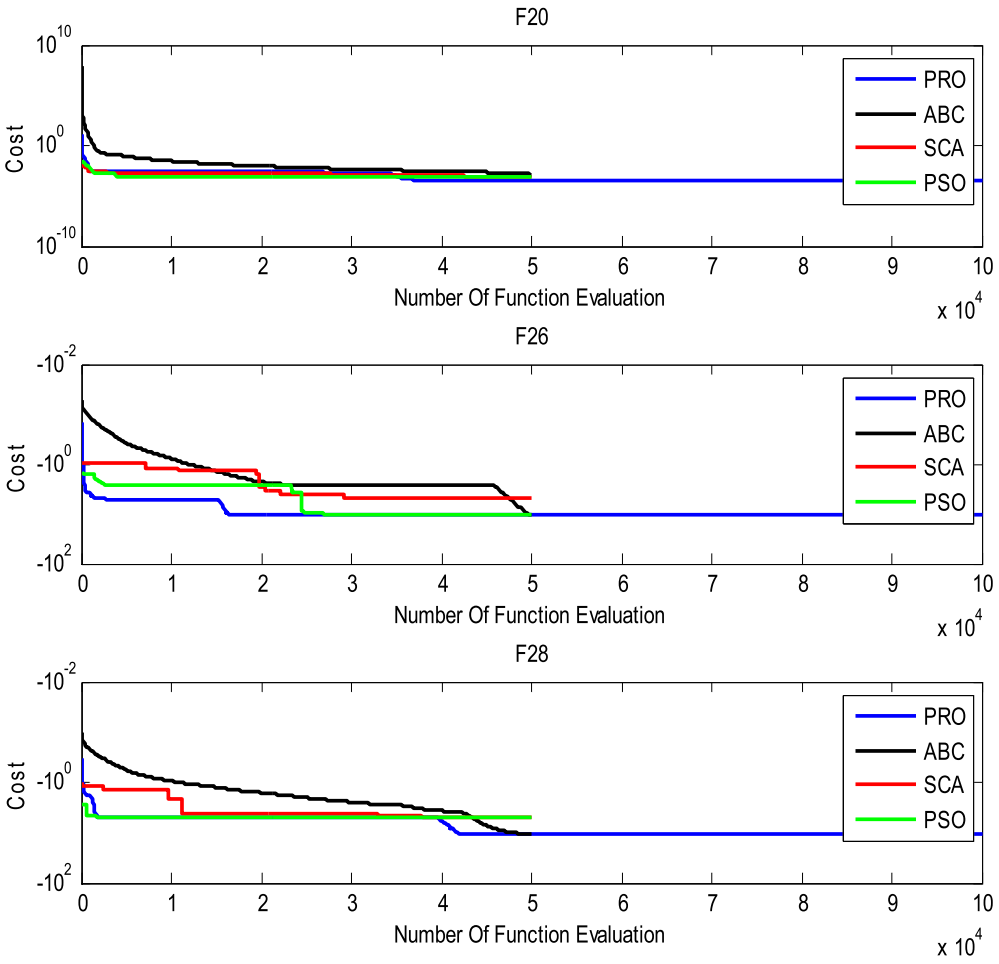

Fig. 6. The convergence of the algorithms on fixed-dimension multimodal functions.

F25 function. The PRO and MFO algorithms have obtained the same average value from F28 function, but it must be noted that the MFO algorithm has a lower STD value than the PRO algorithm. In Hybrid Functions (F29–F33), the PRO algorithm has achieved the best results for all five functions compared to the other comparable algorithms. The PRO algorithm has obtained the mean value and standard deviation (equals 0) for F29 and F30 and F33 functions. GWO and WOA algorithms have achieved very competitive results compared to the PRO algorithm on F29 and F30 functions; although these results are very close to 0, the PRO algorithm has been able to access the value of 0 for these functions. It is worth noting that the PRO algorithm has been tested statistically and for all algorithms and 33 test functions, the P value is calculated, as all the values of P are less than 0.05.

Finally, by examining the results obtained from 2017 functions (F8–F10, F17–F18, F29–F33), it will be clear that the PRO algorithm has obtained the best value within all these functions. Although it was along with the competitive results in some functions, the PRO algorithm generally has gained the best results compared to the other algorithms.

The performance of PRO is evaluated using 18 test functions (unimodal and multimodal) with 300 dimensions to reveal the power of the algorithm in optimization of large scale problems. The results obtained from the PRO algorithm are compared with three algorithms of Section 3.1, and the best algorithm of Section 3.2 (WOA). The results are shown in Table 12. It can be concluded that PRO has the ability to explore and exploit enough to resolve large scale optimization problems.

Regards the results, the PRO algorithm has obtained the best result from 13 functions among 18 of them. The WOA algorithm has achieved the best results within the other five algorithms. The value of PRO algorithm on F1–F4 and F8–F10 functions equals to 0. Although the WOA algorithm has obtained a result which is very close to the PRO algorithm on F1, F2, F8, and F9 functions, the PRO algorithm has obtained the value of 0. In F7, and F12–F14 functions, the PRO algorithm got the second rank in average and does not perform well on F11 function, and for this reason, its rank was in the fifth level. By examining the results obtained from 2017 functions (F8–F10, F17, F18), it can be noted that the PRO algorithm has achieved the best result within the domain of these functions.

3.3. PRO for classical engineering problems

- **Tension/compression spring design problem**

The main objective of this problem is to minimize the cost of constructing the minimization of the fabrication cost of a spring with three parameters (wire diameter (d), mean coil diameter (D) and the number of active coils (N)). Fig. 8 shows the spring and its parameters.

The mathematical equations of this problem are given in Eq. (8).

$$\begin{aligned} &\text{consider } \vec{x} = [x_1, x_2, x_3] = [\text{dDN}] \\ &\text{minimize } f(\vec{x}) = (x_3 + 2)\, x_2 x_1^2, \end{aligned} \tag{8}$$

Subject to:

$$g_1(x) = 1 - \frac{x_2^3 x_3}{71875 x_1^x} \ll 0,$$

$$g_2(x) = \frac{4x_2^2 - x_1 x_2}{12566\,(x_2 x_1^3 - x_1^4)} + \frac{1}{5108 x_1^2} \ll 0,$$

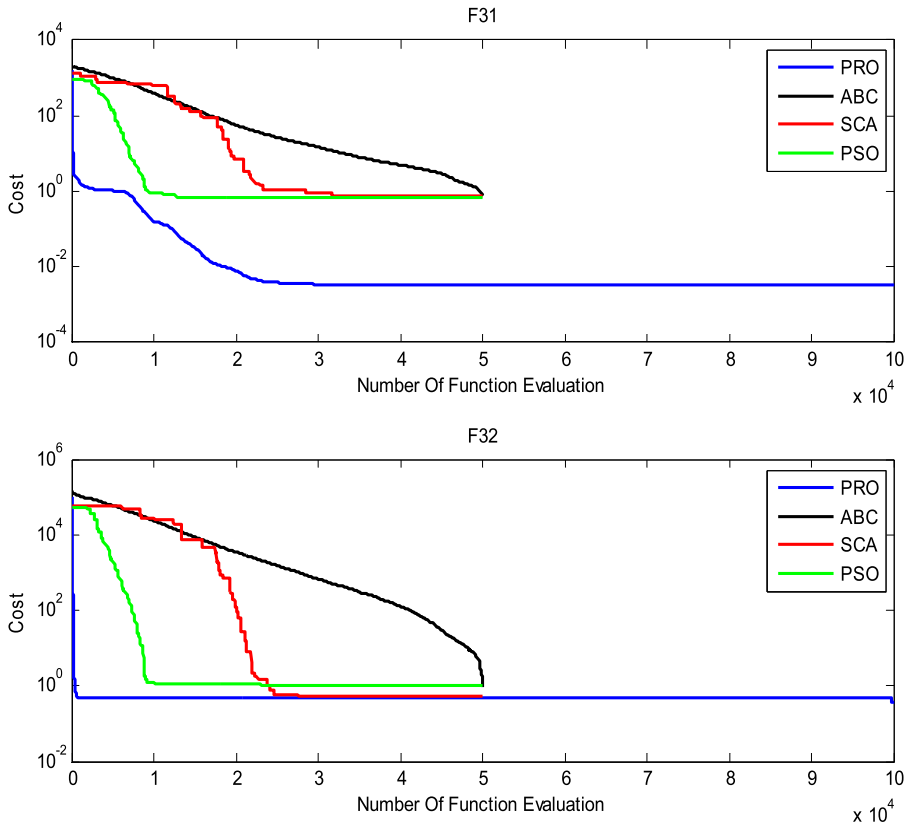

Fig. 7. The convergence of the algorithms on hybrid functions.

Table 7
The results of the optimization algorithms on fixed-dimension multimodal functions.

Function		PRO	SCA	ABC	PSO
F19	AVE	7.47	1.85	**0.99**	**0.99**
	STD	4.75	1.00	1.22E−9	**4.51E−16**
	RANK	4	3	**1**	**1**
F20	AVE	**3.32E−4**	5.62E−4	1.01E−3	2.67E−3
	STD	2.78E−5	3.72E−4	**1.84E−5**	6.2E−3
	RANK	**1**	2	3	4
F21	AVE	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**
	STD	**0**	4.53E−6	4.516E−16	**0**
	RANK	**1**	**1**	**1**	**1**
F22	AVE	**0.397**	0.398	**0.397**	**0.397**
	STD	**1.12E−16**	1.90E−4	**1.12E−16**	**1.12E−16**
	RANK	**1**	4	**1**	**1**
F23	AVE	**3**	**3**	**3**	**3**
	STD	**0**	1.86E−5	1.88E−5	**0**
	RANK	**1**	**1**	**1**	**1**
F24	AVE	**−3.8628**	−3.8562	−3.8627	−0.3004
	STD	1.43E−7	0.003	2.25E−15	**2.25E−16**
	RANK	**1**	3	2	4
F25	AVE	−2.43	−2.95	**−3.3216**	−3.2558
	STD	0.53	0.24	**6.51E−10**	0.08
	RANK	4	3	**1**	2
F26	AVE	**−10.1532**	−3.47	−10.1081	−7.12
	STD	**2.13E−7**	2.82	0.0665	2.9912
	RANK	**1**	4	2	3
F27	AVE	**−10.4029**	−4.97	−10.4025	−9.6060
	STD	**4.01E−5**	1.38	1.1E−3	2.08
	RANK	**1**	4	2	3
F28	AVE	**−10.5363**	−5.18	−10.5358	−7.39
	STD	**4.02E−5**	0.82	1.2E−3	3.91
	RANK	**1**	4	2	3

Table 8
Hybrid functions.

Function	Dim	Range	F_{nmi}
$F_{29}(X) = \sum_{i=1}^{n} (10^6)^{\frac{i-1}{n-1}} X_i^2$	30	[−100,100]	0
$F_{30}(X) = 10^6 X_1^2 + \sum_{i=2}^{n} X_i^2$	30	[−100,100]	0
$F_{31}(X) = \left\lvert \sum_{i=1}^{n} X_i^2 - n \right\rvert^{\frac{1}{4}} + \frac{(0.5 \sum_{i=1}^{n} X_i^2 + \sum_{i=1}^{n} X_i)}{n} + 0.5$	30	[−100,100]	0
$F_{32}(X) = \left\lvert \left(\sum_{i=1}^{n} X_i^2\right)^2 - \left(\sum_{i=1}^{n} X_i\right)^2 \right\rvert^{\frac{1}{2}} + \frac{(0.5 \sum_{i=1}^{n} X_i^2 + \sum_{i=1}^{n} X_i)}{n} + 0.5$	30	[−100,100]	0
$F_{33}(X) = \left[\frac{1}{n-1} \sum_{i=1}^{n-1} \left(\sqrt{s_i} \left(\sin\left(50 s_i^{0.2}\right) + 1 \right) \right) \right]^2, s_i = \sqrt{X_i^2 + X_{i+1}^2}$	30	[−100,100]	0

Table 9
The results of the optimization algorithms on hybrid functions.

Function		PRO	SCA	ABC	PSO
F29	AVE	**0**	0.005	3.14E+3	1.6604E+4
	STD	**0**	0.0053	3.26E+3	3.08E+3
	RANK	**1**	2	**3**	**4**
F30	AVE	**0**	0.001	0.43	5.65E+5
	STD	**0**	7.84E−4	0.17	6.35E+5
	RANK	**1**	2	3	4
F31	AVE	**0.003**	0.87	0.74	0.62
	STD	**8.30E−4**	0.19	0.08	0.18
	RANK	**1**	**4**	**3**	**2**
F32	AVE	**0.27**	0.53	3.38	1.24
	STD	**0.04**	0.10	2.74	0.91
	RANK	**1**	2	**4**	**3**
F33	AVE	**0**	0.98	1.31E+14	3.87E+6
	STD	**0**	0.56	2.11E+14	4.15E+6
	RANK	**1**	**2**	**4**	**3**

Table 10
The results on unimodal and multi-modal functions.

Functions		PRO	GWO (2014)	ALO (2015)	MFO (2015)	DA (2015)	WOA (2016)	SSA (2017)
F1	AVE	**0**	1.76E−69	4.60E−7	1.94E−5	0.27	2.16E−146	9.33E−9
	STD	**0**	3.57E−69	4.61E−7	1.89E−5	0.03	4.82E−146	1.63E−9
F2	AVE	**0**	2.46E−41	34.56	36	0.007	8.19E−110	1.44
	STD	**0**	1.21E−41	47.42	20.73	0.009	1.31E−109	1.14
F3	AVE	**0**	3.65E−19	2.67E+2	2.18E+4	18.94	1.15E+4	41.88
	STD	**0**	7.74E−19	1.16E+2	1.62E+4	0.16	7.57E+3	30.24
F4	AVE	**0**	6.49E−18	7.84	56.90	0.19	48.63	4.60
	STD	**0**	6.86E−18	3.94	3.57	0.14	24.91	2.45
F5	AVE	**5.73E−14**	26.55	2.73E+2	75.34	6.13	26.54	40.79
	STD	**1.99E−13**	1.00	4.43E+2	29.40	1.36	0.45	33.35
F6	AVE	**4.16E−20**	0.29	6.69E−7	9.44E−5	1.88	0.003	8.10E−9
	STD	**7.88E−20**	0.32	4.30E−7	1.58E−4	1.02	0.001	1.82E−9
F7	AVE	**3.43E−05**	4.88E−4	0.04	0.05	0.29	0.0011	0.05
	STD	**2.36E−05**	2.03E−4	0.01	0.01	0.04	0.0012	0.02
F8	AVE	**0**	7.03E−65	81.08	1.02E+4	1.37E+7	2.54E−170	1.06E+3
	STD	**0**	8.30E−65	8.92	5.36E+3	4.85E+6	**0**	1.05E+3
F9	AVE	**0**	1.84E−217	2.13E+9	4.04E+27	4.95E+9	7.16E−248	3.99E+8
	STD	**0**	**0**	2.02E+9	5.44E+27	1.84E+9	**0**	7.47E+8
F10	AVE	**0**	8.22E−65	2.53E−7	0.024	5.20E+3	1.07E−8	1.24E−8
	STD	**0**	8.05E−65	4.29E−7	0.025	2.84E+3	2.39E−8	3.55E−9
F11	AVE	−9.06E+3	−5.42E+3	−5.48E+3	−8.38E+3	−6.63E+3	**−1.20E+4**	−7.47E+3
	STD	944.85	1.31e+3	57.89	441.39	313.53	580.51	401.59
F12	AVE	**0**	3.41E−14	69.58	134.72	18.47	**0**	55.12
	STD	**0**	5.09E−14	25.53	31.86	6.79	**0**	10.01
F13	AVE	**8.88E−16**	1.36E−14	1.79	11.86	4.92	3.01E−15	2.16
	STD	**1.00E−31**	1.97E−15	0.91	10.13	3.57	1.94E−15	0.42
F14	AVE	**0**	0.002	0.01	0.012	0.22	**0**	0.008
	STD	**0**	0.005	0.009	0.004	0.13	**0**	0.005
F15	AVE	**4.48E−21**	0.015	8.65	1.03	2.81	0.001	4.65
	STD	**1.18E−20**	0.017	3.49	1.43	1.05	0.003	3.03
F16	AVE	**2.77E−21**	0.26	1.16E−2	0.005	0.021	0.10	6.70E−10
	STD	**5.37E−21**	0.21	0.01	0.006	0.008	0.14	1.56E−10
F17	AVE	**2.74E−20**	26.87	21.54	4.43E+2	3.59E+6	26.95	1.43E+3
	STD	**6.24E−20**	0.72	1.47	4.24E+2	3.39E+6	0.09	1.67E+3
F18	AVE	**6.59E−21**	1.18	9.14E+2	7.45E+3	9.99E+2	0.78	2.85E+2
	STD	**2.50E−20**	0.11	5.88E+2	1.10E+3	5.57E+2	0.49	1.22E+2

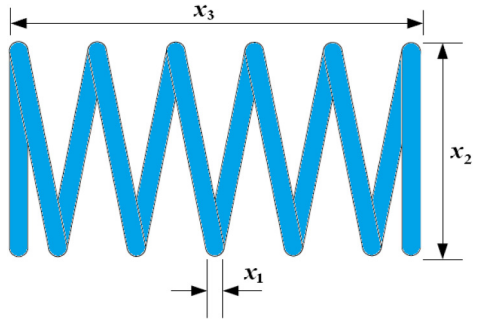

Fig. 8. Tension/compression spring.

$$g_3(x) = 1 - \frac{14045x_1}{x_2^2 x_3} \ll 0,$$

$$g_4(x) = \frac{x_1 + x_2}{1.5} - 1 \ll 0,$$

Variable range:

$$0.05 \ll x_1 \ll 2.00,$$

$$0.25 \ll x_2 \ll 1.30,$$

$$2.00 \ll x_3 \ll 15.0.$$

Many optimization algorithms, including MFO (Mirjalili, 2015c), PSO (He and Wang, 2007), GWO (Mirjalili et al., 2014), GA (Coello Coello, 2000), and WOA (Mirjalili and Lewis, 2016), have solved this problem. Table 13 has compared the results obtained from PRO algorithm with the results obtained from the five algorithms mentioned. The results obtained from these five algorithms are the best results obtained from the papers. PRO algorithm obtained the most accurate results by setting its essential parameter to mutation probability = 0.06, in comparison with other algorithms.

• **Pressure vessel design problem**

A compressed air storage tank with a working pressure of 3000 psi and a minimum volume of 750 ft3. A cylindrical vessel is capped at both ends by hemispherical heads (see Fig. 9). Using rolled steel plate, the shell is made in two halves that are joined by two longitudinal welds to form a cylinder.

The objective is minimizing the total cost, including the cost of the materials forming the welding. The design variables are: thickness x1, thickness of the head x2, the inner radius x3, and the length of the cylindrical section of the vessel x4. The variables x1 and x2 are discrete values which are integer multiples of 0.0625 inch. Then, the formal statement is:

$$\text{consider } \vec{x} = [x_1 x_2 x_3 x_4] = [T_s T_h\ RL], \tag{9}$$

$$\text{minimize } f(\vec{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$$

Subject to:

$$g_1(\vec{x}) = -x_1 + 0.0193x_3 \ll 0,$$

Table 11
The results on fixed dimension multimodal and hybrid functions.

Function		PRO	GWO (2014)	ALO (2015)	MFO (2015)	DA (2015)	WOA (2016)	SSA (2017)
F19	AVE	7.47	**0.99**	2.7326	**0.99**	**0.99**	**0.99**	**0.99**
	STD	4.75	**0**	1.24	1.24E−16	**0**	**0**	**0**
F20	AVE	**3.32E−4**	0.008	9.21E−4	0.002	0.003	5.13E−4	7.57E−4
	STD	**2.78E−5**	0.010	1.04E−4	0.003	5.69E−4	1.22E−4	3.25E−4
F21	AVE	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**
	STD	**0**	**0**	**0**	**0**	**0**	**0**	**0**
F22	AVE	**0.397**	**0.397**	**0.397**	**0.397**	**0.397**	**0.397**	**0.397**
	STD	1.12E−16	8.34E−7	**0**	**0**	5.94E−8	**0**	**0**
F23	AVE	**3**	**3**	**3**	**3**	**3**	**3**	**3**
	STD	**0**	3.36E−6	**0**	**0**	**0**	**0**	**0**
F24	AVE	**−3.8628**	−3.8613	**−3.8628**	**−3.8628**	−3.8627	−3.8621	**−3.8628**
	STD	1.43E−7	0.003	**0**	**0**	7.60E−5	8.82E−4	**0**
F25	AVE	−2.43	−3.2744	−3.2328	−3.2020	−3.1694	**−3.2970**	−3.2024
	STD	0.53	0.06	0.05	**0.002**	4.48E−5	0.05	6.79E−4
F26	AVE	**−10.1532**	−10.1530	−5.7480	−7.2340	−5.0936	−10.1529	−9.1427
	STD	**2.13E−7**	7.17E−5	3.14	2.66	2.43	2.36E−4	2.25
F27	AVE	**−10.4029**	−10.4027	−8.7333	−6.0068	−9.3308	−10.4026	−9.3481
	STD	**4.01E−5**	1.53E−4	3.33	4.03	2.37	3.31E−4	2.35
F28	AVE	**−10.5363**	−10.5361	−8.6040	**−10.5363**	−8.9531	−9.4541	−8.9905
	STD	**4.02E−5**	1.34E−4	3.86	**0**	3.43	2.41	3.45
F29	AVE	**0**	2.70E−67	2.01E+6	8.73E+6	3.87E+7	2.18E−169	3.22E+6
	STD	**0**	3.42E−67	2.41E+6	6.92E+6	2.09E+7	**0**	2.56E+6
F30	AVE	**0**	2.92E−70	4.61E+4	1.49E−4	2.93E+4	3.99E−177	6.82E+3
	STD	**0**	4.99E−70	2.19E+4	1.40E−4	2.72E+4	**0**	4.52E+3
F31	AVE	**0.003**	0.55	1.28	0.54	22.45	0.52	0.59
	STD	**8.30E−4**	0.06	0.79	0.26	3.82	0.12	0.03
F32	AVE	**0.27**	0.47	0.58	0.80	8.47E+2	0.36	0.57
	STD	**0.04**	0.03	0.23	0.41	0.70	0.10	0.28
F33	AVE	**0**	1.396E−6	1.36E+10	0.13	2.11E+12	3.19E−4	3.29E+9
	STD	**0**	1.398E−6	3.82E+9	0.11	2.97E+12	7.03E−4	4.04E+9

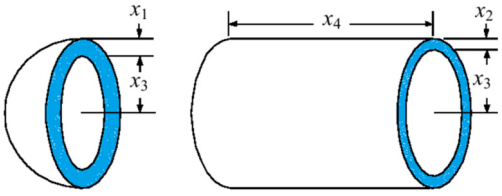

Fig. 9. Pressure Vessel.

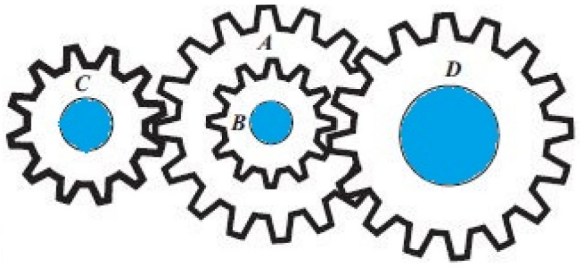

Fig. 10. Gear train design problem.

$$g_2(\vec{x}) = -x_3 + 0.00954x_3 \ll 0,$$

$$g_3(\vec{x}) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1269000 \ll 0,$$

$$g_4(\vec{x}) = x_4 - 240 \ll 0,$$

Variable range:

$$0 \ll x_1 x_2 \ll 99,$$

$$10 \ll x_3 x_4 \ll 200.$$

Many optimization algorithms, including MFO (Mirjalili, 2015c), PSO (He and Wang, 2007), GWO (Mirjalili et al., 2014), and WOA (Mirjalili and Lewis, 2016) have solved this problem. Table 14 shows the comparison of the results obtained from PRO algorithm with the results obtained from other algorithms. The results obtained from these algorithms are the best results from the articles. PRO algorithm, by setting its essential parameter to mutation probability=0.06 has achieved the most accurate result compared to other algorithms.

• **Gear train design problem**

This is a mechanical engineering problem which aims for the minimization of gear ratio for a given set of four gears of a train. The parameters are the number of teeth of the gears, so there is a total of 4 variables for this problem. There is no constraint in this problem, but the ranges of variables are considered as constraints. The overall schematic of the system is illustrated in Fig. 10.

The mathematical equations of this problem are given in Eq. (10).

$$\text{consider } \vec{x} = [x_1 x_2\ x_3\ x_4] = [n_A\ n_B\ n_C\ n_D], \tag{10}$$

$$\text{minimize } f(\vec{x}) = \left(\frac{1}{6.931} - \frac{x_3 x_2}{x_1 x_4}\right)^2,$$

subject to :

$$12 \ll x_1, x_2, x_3, x_4 \ll 60.$$

Many optimization algorithms, including MFO (Mirjalili, 2015c), ABC (Sharma et al., 2012), ALO (Mirjalili, 2015a), and MBA (Sadollah et al., 2013b) have solved this problem. Table 15 shows the comparison of the results obtained from PRO algorithm with the results obtained from other algorithms. PRO algorithm, by setting its essential parameter to mutation probability=0.06 has achieved the equal result compared to other algorithms.

Table 12
The results of algorithms on test functions with 300 dimensions.

Functions		PRO	SCA	ABC	PSO	WOA
F1	AVE	**0**	7.80E+5	8.67E+5	3.95E+3	3.65E−174
	RANK	**1**	4	5	3	2
F2	AVE	**0**	6.13	9.92E+4	56.24	9.18E−113
	RANK	**1**	3	5	4	2
F3	AVE	**0**	2.48E+5	1.01E+7	3.68E+5	5.32E−10
	RANK	**1**	3	5	4	2
F4	AVE	**0**	1.05E+2	98.48	40.14	84.06
	RANK	**1**	5	4	2	3
F5	AVE	**1.15E−4**	4.36E+8	3.98E+9	5.34E+5	2.96E+2
	RANK	**1**	4	5	3	2
F6	AVE	**1.89E−6**	3.13E+4	8.89E+5	3.40E+3	5.30
	RANK	**1**	4	5	3	2
F7	AVE	0.01	2.74E+3	1.88E+4	9.13	**2.80E−4**
	RANK	2	4	5	3	**1**
F8	AVE	**0**	1.11E+3	2.63E+2	3.10E−9	1.92E−173
	RANK	**1**	5	4	3	2
F9	AVE	**0**	8.75	5.66E+26	2.95E+3	3.54E−265
	RANK	**1**	3	5	4	2
F10	AVE	**0**	6.89	2.10E+4	1.25E+4	8.22E−10
	RANK	**1**	3	5	4	2
F11	AVE	−5.25E+4	−5.96E+4	−5.83E+4	−5.84E+4	**−1.22E+5**
	RANK	5	2	4	3	**1**
F12	AVE	2.91E−18	1.32E+2	5.13E+3	1.14E+3	**0**
	RANK	2	3	5	4	**1**
F13	AVE	7.37E−4	38.01	21.07	7.55	**2.66E−15**
	RANK	2	5	4	3	**1**
F14	AVE	2.40E−6	2.44E+2	7.99E+3	32.75	**0**
	RANK	2	4	5	3	**1**
F15	AVE	**4.60E−7**	1.46E+9	1.01E+10	1.02E+4	0.01
	RANK	**1**	4	5	3	2
F16	AVE	**4.51E−6**	2.25E+9	1.78E+10	3.59E+5	2.70
	RANK	**1**	4	5	3	2
F17	AVE	**1.89E−22**	1.32E+3	1.48E+5	2.34E+2	26.79
	RANK	**1**	4	5	3	2
F18	AVE	**3.42E−25**	4.70	3.44E+2	4.95E+3	0.34
	RANK	**1**	3	4	5	2

Table 13
Comparison of results for tension/compression spring design problem.

Algorithm	d	D	N	f
PRO	0.051819	0.359873	11.106372	**0.012665**
MFO (Mirjalili, 2015c)	0.051994	0.364109	10.868421	0.012666
PSO (He and Wang, 2007)	0.051728	0.357644	11.244543	0.012674
GWO (Mirjalili et al., 2014)	0.051690	0.356737	11.288850	0.012666
GA (Coello Coello, 2000)	0.051480	0.351661	11.632201	0.012704
WOA (Mirjalili and Lewis, 2016)	0.051207	0.345215	12.004032	0.012676

Table 14
Comparison of results for pressure vessel design problem.

Algorithm	T_s	T_h	R	L	f
PRO	0.7445	0.4424	38.489983	200.0000	**6050.7134**
MFO (Mirjalili, 2015c)	0.8125	0.4375	42.098445	176.6365	6059.7143
PSO (He and Wang, 2007)	0.8125	0.4375	42.091266	176.7465	6061.0777
GWO (Mirjalili et al., 2014)	0.8125	0.4345	42.089181	176.7587	6051.5639
WOA (Mirjalili and Lewis, 2016)	0.8125	0.4375	42.098269	176.6389	6059.7410

• Three-bar truss design problem

Three-bas truss design problem is another structural optimization problem in the field of civil engineering. The objective is to design a truss with the minimum weight that does not violate the constraints. The most important issue in designing truss is constraints that include stress, deflection, and buckling constraints. Fig. 11 shows the details of this problem.

Table 15
Comparison of results for gear train design problem.

Algorithm	n_A	n_B	n_C	n_D	f
PRO	43	16	19	49	**2.7009 e−12**
MFO (Mirjalili, 2015c)	43	19	16	49	**2.7009 e−12**
ABC (Sharma et al., 2012)	49	16	19	43	**2.7009 e−12**
ALO (Mirjalili, 2015a)	49	16	19	43	**2.7009 e−12**
MBA (Sadollah et al., 2013b)	43	16	19	49	**2.7009 e−12**

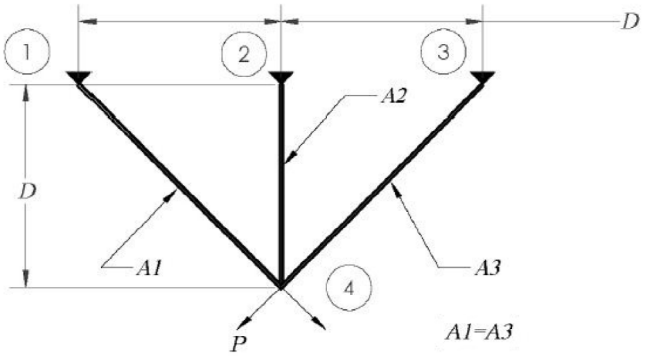

Fig. 11. Three-bar truss.

Table 16
Comparison of results for three-bar truss design problem.

Algorithm	X_1	X_2	F
PRO	0.7886475	0.4083262	263.8958439
GOA (Saremi et al., 2017)	0.7888975	0.4076195	263.8958814
MFO (Mirjalili, 2015c)	0.7882447	0.4094669	263.8959796
PSO-DE (Liu et al., 2010)	0.7886751	0.4082482	**263.8958433**
ALO (Mirjalili, 2015a)	0.7886628	0.4082831	263.8958434
MVO (Mirjalili et al., 2015)	0.7886027	0.4084530	263.8958499

In Eq. (11) this problem is shown.

$$\text{consider } \vec{x} = [x_1 x_2] = [A_1\ A_2], \tag{11}$$

$$\text{minimize } f(\vec{x}) = \left(2\sqrt{2}x_1 + x_2\right) * l$$

subject to :

$$g_1(\vec{x}) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2} p - \sigma \ll 0,$$

$$g_2(\vec{x}) = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2} p - \sigma \ll 0,$$

$$g_3(\vec{x}) = \frac{1}{\sqrt{2}x_2 + x_1} p - \sigma \ll 0,$$

Variable range:

$$0 \ll x_1x_2 \ll 1,\ l = 100, p = 2, \sigma = 2$$

The results obtained from PRO algorithm have been compared with the MFO algorithms (Mirjalili, 2015c), PSO-DE (Liu et al., 2010), ALO (Mirjalili, 2015a) and MVO (Mirjalili et al., 2015) in Table 16. As the results show, PSO-DE algorithm has achieved the best results. PRO algorithm has solved this problem by setting its essential parameter to mutation probability= 0.06.

3.4. PRO for software effort estimation

3.4.1. Use case point

The UCP method is the extension of Function Point method with the benefit of requirement analysis in the object-oriented process. The UCP estimation method was first introduced by Sharma et al. (2012) to predict the size of object-oriented software projects. The UCP is computed by converting the elements of UML use case diagram to size metrics through a well-defined procedure. The estimator must classify the actors in the use case diagram into three categories according to their difficulties: simple, average, and complex. According to that the Unadjusted Actor Weights (UAW) is computed. UAW is calculated by Eq. (12).

$$UAW = 1SA + 2AA + 3CA \tag{12}$$

In this equation, SA, AA and CA are respectively the numbers of simple, average and complex actors. Similarly, the use cases are also classified into three classes (simple, average, and complex) based on the number of transactions mentioned in the use case descriptions. Based on the provided description, unadjusted use case (UUC) is calculated by Eq. (13).

$$UUC = 5SUC + 10AUC + 1 \times CUC \tag{13}$$

In this Equation, SUC, AUC and CUC are respectively the numbers of simple, average and complex use cases. Unadjusted use case points UUCP is calculated by Eq. (14) based on the obtained values of UAW and UCC.

$$UUCP = UAW + UUC \tag{14}$$

UUCP should be adjusting by two groups of factors namely Technical Complexity Adjustment Factor (TCF) and Environmental Adjustment Factor (EF). TCF is computed from a set of 13 technical factors (F_1, F_2, …, F_{13}) that have great influence on project performance. TCF is calculated by Eq. (15).

$$TCF = 0.6 + \left(0.01 \sum_{i=1}^{13} (F_i f w_i)\right) \tag{15}$$

where F_i the value of influence of factor i, and fw_i the weight associated with factor i. EF is computed from a set of eight environmental factors (E_1, E_2, , E_8) that have great effect on productivity. EF is calculated by Eq. (16).

$$EF = 1.4 - \left(0.03 \sum_{i=1}^{8} (E_i ew_i)\right) \tag{16}$$

where E_i the value of influence of factor i, and ew_i the weight associated with factor i. Final value of UCP is calculated by Eq. (17) according to the above-mentioned description.

$$UCP = UUCP \times TCF \times EF \tag{17}$$

3.4.2. Proposed model

In the present paper, the proposed model aims is to weighting environmental factors accurately through PRO algorithm to obtain the most accurate result of evaluation criteria. In this paper, the accuracy of the proposed model is evaluated with respect to four criteria, namely Absolute Error (AE), Mean Absolute Error (MAE), Mean Balanced Relative error (MBRE), and Mean Inverted Balanced Relative Error (MIBRE) which are shown in Eqs. (18) to (21) respectively.

$$AE_i = \left|effort_i - \widehat{effort}_i\right| \tag{18}$$

$$MAE = \frac{\sum_i^n AE_i}{n} \tag{19}$$

$$MBRE = \frac{1}{n} \sum_1^n \frac{AE_i}{\left(\min\left(effort_i, \widehat{effort}_i\right)\right)} \tag{20}$$

$$MIBRE = \frac{1}{n} \sum_1^n \frac{AE_i}{\left(\max\left(effort_i, \widehat{effor}_i\right)\right)} \tag{21}$$

In these equations, $effort_i$ and $\widehat{effort}_i$ are respectively the main and estimation of effort in project i. In this paper also uses two data sets to compare the proposed model with other models. The first dataset contains 45 industrial projects collected from software organization. The architecture of these projects was two-tier and three-tier applications (Nassif et al., 2013). The second dataset contains 65 educational projects that were collected from 4th year and Master's university students. The projects have been designed and coded using UML diagrams and OO programming languages (Nassif et al., 2013). The proposed model in this paper consists of two main stages of training and testing which are described as follows.

3.4.3. Training stage in the proposed model

At this stage of the proposed model, data set is randomly divided into three equal groups (approximately equal). Two groups are selected for training stage and one group for testing stage. This process continues until all three groups are selected for the testing stage. Finally, the average result of three stages is considered as the final result. Each selected project includes eight environmental factors. 9 weights are proposed for selected project by PRO Algorithm. 8 of 9 weights are allocated to 8 environmental factors of this project, and the ninth weight is considered as the project size power. After weighting to environmental factors, the environmental factor multiplication (MEF) is calculated. MEF value is used to calculate the effort and it is calculated by Eq. (22).

$$MEF_i = W_1EF_1 + W_2EF_2 + \cdots + W_8EF_8 = \sum_{i=1}^{8} W_iEF_i \tag{22}$$

In this equation, W_i is the weight of project i, and EF_i is the environmental factor i of target project. Effort of this project should be

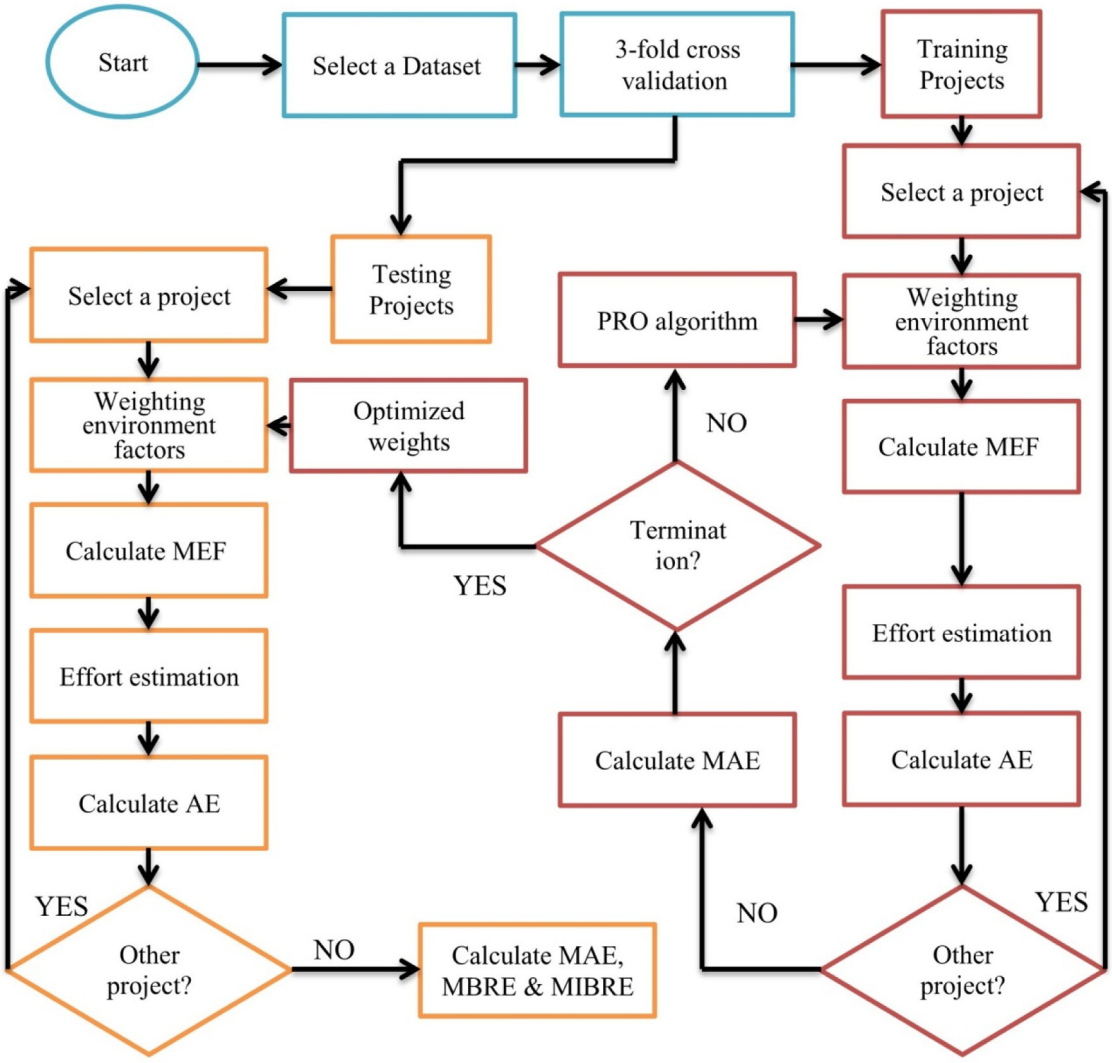

Fig. 12. Flowchart of proposed model.

calculated after calculating MEF of project. Effort of each project is calculated by Eq. (23).

$$\text{Effort}_i = \text{Size}^{W_9} \text{MEF}_i \quad (23)$$

In this equation, SIZE refers to size of project i; and W_9 is the ninth weight proposed by PRO Algorithm. In this section, the ninth weight is selected as the project size power because a change in project size can have a profound effect on estimation of its effort. AE error of a project should be measured after calculating the project effort and this will continue until there is not any other project for training. Error of cost function should be calculated at training stage by measuring AE error of all projects. The purpose of this paper is to minimize the MAE in any iteration of the PRO algorithm. The best proposed weights in a certain iteration (iteration with the minimum error) are saved as the final result after calculating cost function value of training stage in the case that the termination conditions (ending number of iterations) are met for PRO Algorithm.

3.4.4. Testing stage in the proposed model

At this stage, one of testing projects is selected, and like the training stage, 8 weights are considered for environmental factors and one weight for project size power. On the contrary, the allocated weights to testing projects are the best weights which are obtained from the training stage. MEF of target project is calculated after allocating weights, and then the amount of effort is measured by Eq. (23). AE error can be calculated for this project according to estimated effort and actual amount of effort. Other values of testing stage performance criteria can be calculated after calculation of AE for all projects. Fig. 12 shows the general flowchart of proposed model for training and testing stages.

3.4.5. Results

In this section, we compared results of proposed model with results of other models and algorithms. The present paper compared the proposed model with models Karner (1993), Schneider and Winters (S&W) (Schneider and Winters, 2001) and Nassif et al. (2013). The proposed model was also compared with PSO, ABC and SCA algorithms for more accurate evaluation. In fact, three mentioned algorithms were used in the proposed model of paper instead of PRO algorithm.

To obtain the results of the parameters of the optimization algorithms, the initial settings are required to be described as follows:

For the maximum iteration parameter, three values of 500 and 1000 were considered and it should mention that all algorithms have achieved the most optimal value of 500 as the output. For the initial population parameter, values of 30 and 50 are considered, and the decision of 30 for all algorithms results in an optimal output. In the PRO algorithm, the mutation probability parameter [0.05, 0.09] is considered where the value of 0.05 was the most appropriate. The

Table 17
Obtained results from different models.

Model	Dataset1			Dataset2		
	MAE	MBRE	MIBRE	MAE	MBRE	MIBRE
Karner	6120.62	28.22	20.72	329.04	23.18	17.40
S&W	3893.73	21.46	16.67	446.42	30.36	19.70
Nassif	4419.96	25.00	18.47	325.64	24.28	17.78
PRO-UCP	**3135.10**	**16.53**	**13.28**	**305.16**	**21.14**	**15.23**
PSO-UCP	3987.32	22.35	17.23	326.88	24.91	17.96
ABC-UCP	4011.53	23.87	17.92	340.12	25.68	18.28
SCA-UCP	4318.51	24.16	18.01	346.23	26.03	18.72

parameters C1 and C2 of the PSO algorithm are considered in Holland (1992) and Holland and Reitman (1977), which are an appropriate value for both parameters as well as a value of 1 for the W parameter is chosen. In the ABC algorithm, for the limit parameter the interval value [100,200] is selected, which eventually selection of 100 caused the optimal solution for this algorithm. In Table 17, the obtained results of proposed model are compared with other models. As shown, the proposed model with PRO algorithm has better performance on both datasets. In fact, the PRO algorithm could improve performance of proposed model and obtain the best results for three assessment criteria compared to other models by proposing the most reasonable weights.

4. Discussion

To sum up, the proposed algorithm was evaluated using 33 mathematical benchmark functions (2005–2017), 4 engineering classical problems and a software effort estimation problem. The comparison of PRO and other algorithms shows the superiority of PRO in terms of exploration and exploitation parameters. It must be noted that PRO produces considerably acceptable results on 2017 benchmark functions as compared to 2005 functions. The other point is the capability of PRO to generate accurate results in engineering problems. This type of problems is considered as complicated optimization problems which need high optimization power to reach acceptable results. According to the comparisons, PRO outperforms the other algorithms in engineering problems. Finally, software effort estimation is an important problem in the field of software engineering which has attracted the attention of researchers during the recent years and many optimization algorithms have been employed to solve this problem. The performance of PRO is more accurate in estimating the effort of software projects as compared to other algorithms. Therefore, three different types of comparisons show the acceptable performance of PRO in real world and the promising results obtained from simulation can be generalized due to the wide domain of comparisons.

5. Conclusion

In this paper, the PRO algorithm is proposed as a novel optimization algorithm to solve optimization problems. The proposed algorithm was designed based on the poor and the rich populations and was inspired by the fact that (1) every member of the poor population tries to learn from the rich and improve the economic status and reduce the class gap, and (2) every member of the rich population observes the poor and tries to gain wealth and increase the class gap. The performance of PRO was evaluated using 33 unimodal, multi-modal, fixed-dimension multi-modal and hybrid test functions. The PRO algorithm was compared to PSO, ABC, SCA, ALO, GWO, MFO, WOA, DA and SSA algorithms. The results obtained from PRO on the test functions showed that this algorithm has found better answers compared to the other algorithms in most cases. In addition, the convergence trend diagrams showed that the PRO algorithm converges in fewer number of function evaluation compared to the other algorithms. The statistical test was also conducted which confirmed the significance of difference existing between PRO and the other algorithms. The performance of PRO was evaluated using large scale problems, which showed its acceptable capability as compared to the other algorithms. Finally, for a more accurate evaluation of PRO algorithm, this algorithm has solved four classical engineering problems. The results obtained from these four problems show that PRO algorithm has found optimal parameters for the variables of these problems by careful managing of the conditions of these problems. In total, results show that the PRO algorithm is an appropriate choice to solve the optimization problems. The extensive evaluation domain and the promising results indicate that PRO is capable of providing acceptable results in a wide range of problems. As the future work, we are going to use PRO in real applications and try to extend its usage domain.

References

Askarzadeh, A., Rezazadeh, A., 2012. A new heuristic optimization algorithm for modeling of proton exchange membrane fuel cell: bird mating optimizer. Int. J. Energy Res..

Atashpaz-Gargari, E., Lucas, C., 2007. Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In: Pro- Ceedings of the 2007 IEEE Congress on Evolutionary Computation, CEC. pp. 4661–4667.

Coello Coello, C.A., 2000. Use of a self-adaptive penalty approach for engineering optimization problems. Comput. Ind. 41, 113–127.

Cuevas, E., Cienfuegos, M., Rojas, R., Padilla, A., 2015. A computational intelligence optimization algorithm based on the behavior of the social-spider. In: Computational Intelligence Applications in Modeling and Control. In: Studies in Computational Intelligence, vol. 575, pp. 123–146.

Dai, C., Zhu, Y., Chen, W., 2007. Seeker optimization algorithm. In: Computational Intelli-Gence and Security. Springer, pp. 167–176.

Derrac, J., Garca, S., Molina, D., Herrera, F., 2011. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol. Comput. 1, 3–18.

Dorigo, M., Birattari, M., Stutzle, T., 2006. Ant colony optimization. IEEE Comput. Intell. 1, 28–39.

Eita, M.A., Fahmy, M.M., 2010. Group counseling optimization: a novel approach. In: Bramer, M., Ellis, R., Petridis, M. (Eds.), Research and Development in Intelligent Systems XXVI. Springer, London, pp. 195–208.

Eita, M.A., Fahmy, M.M., 2014. Group counseling optimization. Appl. Softw. Comput. 22, 585–604.

Gandomi, A.H., Alavi, A.H., 2012. Krill Herd: a new bio-inspired optimization algorithm. Commun. Nonlinear Sci. Numer. Simul. 17 (12), 4831–4845.

Geem, Z.W., Kim, J.H., Loganathan, G., 2001. A new heuristic optimization algorithm: harmony search. Simulation 76, 60–68.

Glover, F., 1989. Tabu search –Part I. ORSA J. Comput. 1, 190–206.

Glover, F., 1990. Tabu search –Part II. ORSA J. Comput. 2, 4–32.

Gupta, S., Deep, K., 2018. A novel Random Walk Grey Wolf Optimizer. Swarm Evol. Comput. Accepted Manuscript.

He, Q., Wang, L., 2007. An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Eng. Appl. Artif. Intell. 20, 89–99.

Heidari, A.A., Mirjalili, S.A., Faris, H., Aljarah, I., Mafarja, M., Chen, H., 2019. Harris hawks optimization: Algorithm and applications. Future Gener. Comput. Syst. Accepted Manuscript.

Holland, J.H., 1992. Genetic algorithms. Sci. Am. 267, 66–72.

Holland, J.H., Reitman, J.S., 1977. Cognitive systems based on adaptive algorithms. ACM SIGART Bull. 49.

Hosseinabadi, A.A.R., et al., 2018. Extended genetic algorithm for solving open-shop scheduling problem. Soft Comput. 1–18.

Jamrus, T., Chien, C.-F., 2016. Extended priority-based hybrid genetic algorithm for the less-than-container loading problem. Comput. Ind. Eng. 96, 227–236.

Karaboga, D., Basturk, B., 2007. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Global Optim. 39, 459–471.

Karner, G., 1993. Resource estimation for objectory projects. Object Syst. 1–9.

Kaveh, A., 2014. Colliding bodies optimization. In: Advances in Metaheuristic Algorithms for Optimal Design of Structures. Springer, pp. 195–232.

Kaveh, A., Bakhshpoori, T., 2016. Water Evaporation Optimization: A novel physically inspired optimization algorithm. Comput. Struct. 167, 69–85.

Kaveh, A., Farhoudi, N., 2013. A new optimization method: dolphin echolocation. Adv. Eng. Softw. 59, 53–70.

Kaveh, A., Mahdavi, V., 2014. Colliding bodies optimization: a novel meta-heuristic method. Comput. Struct. 139, 18–27.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks. pp. 1942–1948.

Kumar, M., Kulkarni, A.J., Satapathy, S.C., 2018. Socio evolution & learning optimization algorithm: A socio-inspired optimization methodology. Future Gener. Comput. Syst. 81, 252–272.

Li, X., 2003. A New Intelligent Optimization-Artificial Fish Swarm Algorithm. (Doctor thesis). Zhejiang University of Zhejiang, China.

Liu, H., Cai, Z., Wang, Y., 2010. Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. Appl. Softw. Comput. 10, 629–640.
Mirjalili, S., 2015a. The ant lion optimizer. Adv. Eng. Softw. 83, 80–98.
Mirjalili, S., 2015b. Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput. Appl..
Mirjalili, S., 2015c. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowl.-Based Syst. 89, 228–249.
Mirjalili, S., 2016. SCA: A Sine Cosine Algorithm for solving optimization problems. Knowl.-Based Syst. 96, 120–133.
Mirjalili, S., Lewis, A., 2016. The whale optimization algorithm. Adv. Eng. Softw. 95, 51–67.
Mirjalili, S., Mirjalili, S.M., Hatamlou, A., 2015. Multi-Verse Optimizer: a nature-inspired algorithm for global optimization. Neural Comput. Appl..
Mirjalili, S., Mirjalili, S.M., Lewis, A., 2014. Grey wolf optimizer. Adv. Eng. Softw. 69, 46–61.
Mirjalilia, S., Saremia, S., Mirjalili, S.M., Coelhod, L., 2016. Multi-objective grey wolf optimizer : A novel algorithm for multi-criterion optimization. Expert Syst. Appl. 47, 106–119.
Nassif, A.B., Ho, D., Capretz, L.F., 2013. Towards an early software estimation using log-linear regression and a multilayer perceptron model. J. Syst. Softw. 86 (1), 144–160.
Oftadeh, R., Mahjoob, M.J., Shariatpanahi, M., 2010. A novel meta-heuristic optimization algorithm inspired by group hunting of animals: hunting search. Comput. Math. Appl. 60, 2087–2098.
Rao, R.V., Savsani, V.J., Vakharia, D.P., 2011. Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. Comput. Aided Des. 43, 303–315.
Sadollah, A., Bahreininejad, A., Eskandar, H., Hamdi, M., 2013a. Mine blast algorithm: a new population based algorithm for solving constrained engineering optimization problems. Appl. Softw. Comput. 13, 2592–2612.
Sadollah, A., Bahreininejad, A., Eskandar, H., Hamdi, M., 2013b. Mine blast algorithm: a new population based algorithm for solving constrained engineering optimization problems. Appl. Softw. Comput. 13, 2592–2612.
Salido, M.A., et al., 2016. A genetic algorithm for energy-efficiency in job-shop scheduling. Int. J. Adv. Manuf. Technol. 85, 1303–1314.
Samareh Moosavi, S.H., Khatibi Bardsiri, V., 2017. Satin bowerbird optimizer: A new optimization algorithm to optimize ANFIS for software development effort estimation. Eng. Appl. Artif. Intell. 60, 1–15.
Saremi, S., Mirjalili, S.A., Lewis, A., 2017. Grasshopper Optimization Algorithm: Theory and application. Adv. Eng. Softw. 105, 30–47.
Schneider, G., Winters, J.P., 2001. Applying Use Cases: A Practical Guide, second ed. Addison-Wesley.
Shadravan, S., Naji, H.R., Bardsiri, V.K., 2019. The Sailfish Optimizer: A novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems. Eng. Appl. Artif. Intell. 80, 20–34.
Sharma, T.K., Pant, M., Singh, V., 2012. Improved local search in artificial bee colony using golden section search. arXiv preprint arXiv:1210.6128.
Shiqin, Y., Jianjun, J., Guangxing, Y., 2009. A dolphin partner optimization. In: Proceed- Ings of the WRI Global Congress on Intelligent Systems, GCIS'09. pp. 124–128.
Yang, X.-S., 2010a. Firefly algorithm, stochastic test functions and design optimization. Int. J. Bio-Inspir. Comput. 2, 78–84.
Yang, X.-S., 2010b. A new metaheuristic bat-inspired algorithm. In: Proceedings of the Workshop on Nature Inspired Cooperative Strategies for Optimization (NICSO 2010). pp. 65–74.
Yang, X.-S., Deb, S., 2009. Cuckoo search via Lévy flights. In: Proceedings of the World Congress on Nature & Biologically Inspired Computing, NaBIC. pp. 210–214.
Yazdani, M., Jolai, F., 2016. Lion Optimization Algorithm (LOA): A nature-inspired metaheuristic algorithm. J. Comput. Des. Eng. 3, 24–36.