IJE TRANSACTIONS A: Basics Vol. 33, No. 10, (October 2020) 1924-1938

International Journal of Engineering

Journal Homepage: www.ije.ir

Projectiles Optimization: A Novel Metaheuristic Algorithm for Global Optimization

M. R. Kahrizi*, S. J. Kabudian

Department of Computer Engineering and Information Technology, Razi University, Kermanshah, Iran

PAPER INFO

Paper history:
Received 27 November 2019
Received in revised form 13 March 2020
Accepted 11 June 2020

Keywords:
Global Optimization
Metaheuristic Optimization Algorithm
Population-based Algorithm
Stochastic Optimization

ABSTRACT

Metaheuristic optimization algorithms are a relatively new class of optimization algorithms that are widely used for difficult optimization problems in which classic methods cannot be applied and are considered as known and very broad methods for crucial optimization problems. In this study, a new metaheuristic optimization algorithm is presented, the main idea of which is inspired by models in kinematics. This algorithm obtains better results compared to other optimization algorithms in this field and is able to explore new paths in its search for desirable points. Hence, after introducing the projectiles optimization (PRO) algorithm, in the first experiment, it is evaluated by the determined test functions of the IEEE congress on evolutionary computation (CEC) and compared with the known and powerful algorithms of this field. In the second try out, the performance of the PRO algorithm is measured in two practical applications, one for the training of the multi-layer perceptron (MLP) neural networks and the other for pattern recognition by Gaussian mixture modeling (GMM). The results of these comparisons are presented in various tables and figures. Based on the presented results, the accuracy and performance of the PRO algorithm are much higher than other existing methods.

doi: *10.5829/ije.2020.33.10a.11*

1. INTRODUCTION

Optimization is a procedure for reaching a state in which the problem has a certain advantage and improvement over other states. In other words, the purpose of optimization is to find a point in the problem space where the measure of fitness is maximized. The problem space is the range of values that exist for each dimension of the problem, and points can be selected from these intervals. The criterion for the fitness of points in the problem space varies according to the nature of the problem. If the type of our problem is of the cost or error, the point where the cost (or error) function has the least value is more fit than other points. This type of optimization is also called minimization. On the other hand, there are some problems in which the optimization is to find a point in the problem space where the value of its objective function is maximum. These problems in the field of optimization are called maximization.

Metaheuristic optimization algorithms are a new class of optimization algorithms that are applied to optimization problems in which classic methods cannot be used and are considered as well-known methods for difficult optimization problems. Metaheuristic algorithms are often inspired by nature. Also, in metaheuristic algorithms, many steps are taken at random. For example, several points in the problem space are randomly selected, and they undergo a series of changes during the execution of the algorithm based on the random mechanism of each algorithm, which ultimately leads to access to the optimal global point.

The rest of this paper is organized as follows. The effective methods and existing studies in the field of metaheuristic optimization are reviewed in Section 2. The proposed algorithm is described in Section 3. The comparison of the proposed method with other algorithms and results are discussed in Section 4. Finally, in Section 5, the conclusions are presented.

*Corresponding Author Email: *kahrizi.mr@gmail.com* (M. R. Kahrizi)

Please cite this article as: M. R. Kahrizi, S. J. Kabudian, Projectiles Optimization: A Novel Metaheuristic Algorithm for Global Optimization, International Journal of Engineering (IJE), IJE TRANSACTIONS A: Basics Vol. 33, No. 10, (October 2020) 1924-1938

2. LITERATURE REVIEW

Metaheuristic optimization algorithms are classified into different groups. References [1, 2] can be referred to for more information about optimization algorithm classification. Here, the purpose is to review some of the known and effective algorithms in the development of the projectiles optimization (PRO) algorithm, and in this section, a specified classification is not followed. Also, the available surveys, such as [3-5] can be referred to for more comprehensive information.

One of the oldest and most known optimization algorithms are genetic algorithms (GA). These algorithms are classified as evolutionary algorithms because they use evolutionary computation. The genetic algorithm is a general application algorithm and can be applied in various forms due to its properties. These properties refer to the difference in the chromosome representation, various procedures of parent selection, and different methods in crossover and mutation. The methods of parent selection in the genetic algorithm have been compared in [6, 7]. Different genetic methods with a focus on new subjects were studied in [8]. However, different types of genetic algorithms have been presented over time, and in [9-12] the hybrid type, multi-objective, parallel, and the new version of this algorithm were presented, respectively. Furthermore, reference [13] can be referred to for more information.

Another old algorithm in the optimization field is the simulated annealing (SA) algorithm [14, 15], in which the Metropolis algorithm [16] has been used. Similar to other algorithms, the various versions of the SA algorithm have been proposed, such as the microcanonical annealing (MA) algorithm [17] in which the Creutz algorithm has been utilized instead of the Metropolis algorithm. In [18], another kind of SA algorithm has been proposed that was called threshold accepting (TA). The comparison between this algorithm and the SA algorithm can be observed in [19]. Noised method (NM) is another version of the SA algorithm. This algorithm and its various versions along with their survey have been studied in [20-23]. Another version of the SA algorithm for continuous spaces can be observed in [24]. In [25], the hybrid version of the SA algorithm has been presented for performance improvement in which the SA algorithm has been combined with the GA. Furthermore, the papers [26-29] can be referred to as a comprehensive history.

Ant colony optimization (ACO) is another optimization algorithm, which has been inspired by the ant colony's motion and is classified under swarm intelligence (SI) based algorithms. Reference [30] can be referred to for more information. ACO algorithms have different classes. For example, in [31], the hybrid type of the ACO algorithm has been presented. References [32-35] can be referred to as examples, the development in the ACO algorithm, and the applicability of the ACO algorithm.

Particle swarm optimization (PSO) algorithm is one of the well-known and powerful optimization algorithms that is inspired by a type of motion based on swarm intelligence. This algorithm is attractive due to its desirable performance, and a great amount of research has been performed on it. Various kinds of PSO algorithms have been presented over time. In [36-38], the topology, structure, and the ideas available in PSO have been discussed. Early convergence and hence getting stuck in local optimum points are drawbacks of the PSO algorithm. In [39-44], an attempt was made to resolve these weaknesses. In [45], a kind of PSO algorithm has been presented for discrete spaces. In [46], another version of the PSO algorithm has been presented for dynamic spaces. In [47], the parallel mode of PSO has been examined. In [48], a multi-objective type of algorithm has been discussed. Other various kinds of PSO algorithms can be followed in [49-51], which have been discussed in conjunction with algorithms and other methods for productivity improvement. The particle swarm algorithm has been used in wide fields due to its generality. The details and statistics of these fields have been given in [52]. Also, papers [53-57] can be referred to for more information about some applications and other hybrid versions of the PSO algorithm and their surveys.

The differential evolution (DE) algorithm [58] is another powerful optimization algorithm classified as an evolutionary algorithm. The DE algorithm is a simple algorithm in the optimization field. As with other algorithms, different types of DE algorithms have been proposed. In [59, 60], the multi-objective models of the DE algorithm have been discussed and used. In [61], the differences between PSO and DE algorithms have been studied. Another version of the DE algorithm has been introduced in [62]. The hybrid versions of the DE algorithm have been given in [63, 64]. Moreover, in [65-68], some methods and applications have been presented in which the DE algorithm determines its parameters in a self-adaptive manner.

3. PROJECTILES OPTIMIZATION ALGORITHM

In the PRO algorithm [69], we have attempted to double the accuracy and power. At PRO, we took advantage form the innovations and pros of other algorithms to avoid reinventing the wheel. This algorithm is inspired by the projectile motion in physics and is governed by its laws.

The basic idea behind the PRO algorithm is derived from missiles launches. Since missiles must have the highest accuracy and least error in targeting, the process of moving

these projectiles can be considered as a model for optimization.

As with the laws of physics related to this type of motion, we are faced with a set of parameters that control the motion of the projectiles. Such as acceleration of gravity in the launch environment, initial projectile velocity, launch angle, and intrusive forces such as friction. In the following the method of quantifying and calculating these parameters is described in the projectile algorithm step by step.

Based on the pseudo-code presented in Figure 1, in the PRO algorithm similar to the other metaheuristic optimization algorithms, the first part is the initialization, in which all of the variables involved in the algorithm are introduced and initialized. After initializing the variables, to create each member of the population, which is called "projectile" here, random numbers with the uniform probability distribution are used in a determined range for each problem. Then, the cost of each member is calculated according to the considered cost function. Then, their maximum and minimum value are obtained.

In the following, the main body of the algorithm is studied, which is inside the main loop. In this part of the algorithm, we use some of the principles, properties, and quantities of physics, such as the projectile motion rules, velocity, and gravitational acceleration. In the main loop of the algorithm, the members are sorted based on their costs in such a way that a member with the minimum cost is ranked as "first," and a member with the maximum cost is ranked as the "last." Then, the fitness of each member of the population is calculated by Equation (1), which results in a number in the range of (0, 1].

$$F(x_i) = \frac{N-(Rank(x_i)-1)}{N} \quad (1)$$

In Equation (1), N denotes the total number of members (projectiles) and $Rank(x)$ having a value in the range of $\{1,2,\ldots, N\}$, determines the rank of each member i among all the members that is calculated based on $C(x_i)$ sorting, and $C(x_i)$ is the cost of member i that can be obtained through using the considered function (cost function) for optimization.

After calculating the fitness $F(x)$ for each member of the population, this fitness is used to determine the projectile launch angle according to Equation (2). It should be noted that as can be observed in Table 1, we have empirically considered the minimum and maximum angles tantamount to $45°$ and $84°$, respectively.

$$\theta(x_i) = \theta_{\min} + F(x_i)(\theta_{\max} - \theta_{\min}) \quad (2)$$

In addition, to determine the gravitational acceleration for each projectile, fitness $F(x)$ is used according to Equation (3). Here, we have experimentally considered the maximum and minimum values of the gravitational acceleration equal to 10 m/s^2 and 1 m/s^2, respectively.

$$G(x_i) = G_{\min} + F(x_i)(G_{\max} - G_{\min}) \quad (3)$$

In Equations (2) and (3) all parameters are determined at the beginning of the algorithm, at the initialization section, except $F(x)$ that is in the main loop of the algorithm and it is calculated separately for each member i, and in each iteration, it has a different value. The values given to the used parameters in the PRO algorithm are presented in Table 1.

PRO Algorithm

Initializing the projectile population (N) with random position, and velocity, gravitation, theta and change probability (CP) of each dimension (D) of the projectiles for moving in the search space;
Evaluating the cost $C(x_i)$ for each projectile i;
Evaluating the rank $Rank(x_i)$ for each projectile i by sorting the costs $C(x)$;

For $n = 1,\ldots, N_{iter}$ *(number of iterations)* **do**

For $i = 1,\ldots,N$*(number of projectiles)* **do**

Evaluate the fitness $F(x_i)$ using

$F(x_i) = (N - (Rank(x_i) - 1)) / N$;

Update the theta $\theta(x_i)$ using

$\theta(x_i) = \theta_{\min} + F(x_i)(\theta_{\max} - \theta_{\min})$;

Update the gravitation $G(x_i)$ using

$G(x_i) = G_{\min} + F(x_i)(G_{\max} - G_{\min})$;

Update the velocity $V(n)$ using

$V(n) = V_{\max} - ((n-1)/(N_{iter} - 1))(V_{\max} - V_{\min})$;

For $d = 1,\ldots,D$*(number of dimensions)* **do**

Create random numbers $r_{i,d}, r2_{i,d}$ by a uniform distribution $U(0,1)$;

Evaluate the horizontal range $R(x_{i,d})$ using

$R(x_i) = V^2(n)\sin(2\theta(x_i))/G(x_i)$ and multiplied by $r_{i,d}$;

Create the new projectile $x2_{i,d}$ randomly using a normal distribution $N(x_{i,d}, R^2(x_{i,d}))$ with CP probability for each dimension,

$$x2_{i,d} = \begin{cases} N(x_{i,d}, R^2(x_{i,d})) & if \quad r2_{i,d} \le CP \\ x_{i,d} & otherwise \end{cases};$$

End

Evaluate the cost $C2(x2_i)$ for each new projectile i;

End

Sort the x and $x2$ by costs of them (C, $C2$) and remove the worse half of them;

Evaluate the rank $Rank(x_i)$ for each projectile i that remain from competitive exclusion by sorting costs;

End

Figure 1. PRO algorithm pseudo code

TABLE 1. The parameters and their determined values in the PRO algorithm which are the same and constant for all test functions

Symbol	Quantity	Value
N	Number of projectiles	D
N_{iter}	Number of iterations in each run	10E+4
D	Number of problem's dimensions	10, 50
$MaxFE$	Maximum function evaluation	10E+4 × D
$LimUp$	The upper limit of the search space	100
$LimLo$	The lower limit of the search space	-100
G_{max}	Maximum value of gravitational acceleration	10 m/s^2
G_{min}	Minimum value of gravitational acceleration	1 m/s^2
θ_{max}	Maximum initial launch angle in degree	84°
θ_{min}	Minimum initial launch angle in degree	45°
CP	The Change Probability of each dimension in each projectile	0.1
V_{max}	Maximum value of initial velocity	$\sqrt{Limup - LimLo}$ m/s
V_{min}	Minimum value of initial velocity	V_{max} × 0.1 m/s

Parameters are initialized in such a way to provide similar and impartial conditions for the comparison of this algorithm with other algorithms. These conditions have been presented in [70] and are not the best condition for the PRO algorithm.

After determining the parameters of the launch angle and the gravitational acceleration, the initial velocity of the projectiles should be calculated. This value is constant here and similar for all the projectiles in each iteration. And in the first iteration, the initial velocity is equal to the predetermined value V_{max}. In each iteration of the algorithm, an amount of the projectiles' initial speed is reduced based on Equation (4) so that in the last iteration, the initial speed is reached the predetermined value V_{min}.

However, to determine the initial values of the variables and update their values in this algorithm, other methods appropriate to the problem conditions may be applied. For example, the Euclidean distance of each projectile to the best projectile (member with the lowest cost) can be used to determine the fitness level, launch angle, gravitational acceleration, and initial velocity.

$$V(n) = V_{\max} - \frac{n-1}{N_{\text{iter}} - 1}\left(V_{\max} - V_{\min}\right) \quad (4)$$

where n denotes the current iteration of the algorithm, and the minimum and maximum of its values are 1 and N_{iter}, respectively.

After determining the required parameters, the horizontal range of each projectile should be calculated that is calculated from Equation (5) based on the physics laws governing the projectiles' motion.

$$R(x_i) = \frac{V^2(n)\, sin(2\theta(x_i))}{G(x_i)} \quad (5)$$

As can be seen in Figure 2, this figure shows the variations of the range (it is considered here as a movement in the problem space) with the variations in the launch angle. In this figure, the initial velocity and the gravitational acceleration in the launch site are considered to be equal to 20 m/s and 10 m/s^2, respectively. In addition, in Figures 3 and 4, the projectiles range amount with respect to the variations in the gravitational acceleration in the launch site, and the projectiles range amount with respect to the variations in the launch initial velocity are depicted respectively in which the launch angle is considered equal to 45°.

After determining the horizontal range value $R(x_i)$ for all members of the algorithm population, these values are multiplied by a random array in the range (0, 1) with a uniform distribution. These random numbers are different for each member and each dimension. This can be assumed as the effect of the disruptive forces on the projectiles' motion. However, we decide to apply the effect of the disruptive forces in two stages, and the first phase is applied in this section.

Then, we determine in which dimension the member (projectile) can be altered, in other words, in which dimension of the problem space the projectile can be moved. This action is performed randomly and with CP probability for each dimension of the members. CP determines the change probability of each member in every dimension that is initialized at the beginning of the algorithm.

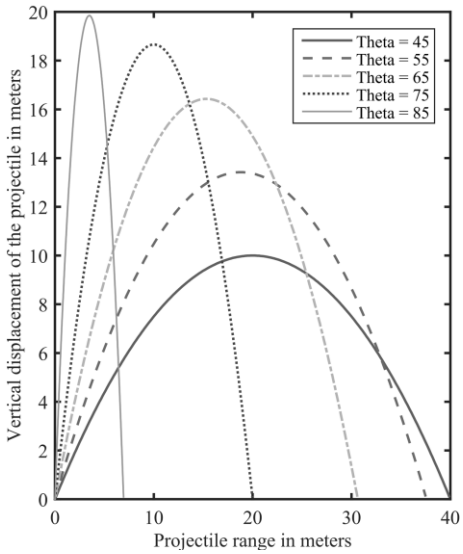

Figure 2. The projectile range with respect to the variation of the launch angle, with the initial velocity of 20 m/s and the gravitational acceleration of 10 m/s^2

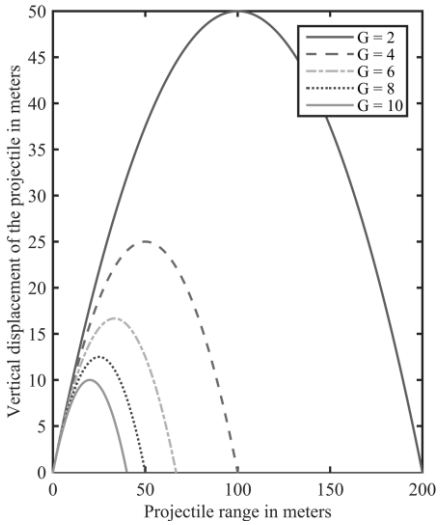

Figure 3. The projectile range with respect to the variation of the gravitational acceleration in the launch point, with the initial velocity of 20 m/s and the launch angle of 45°

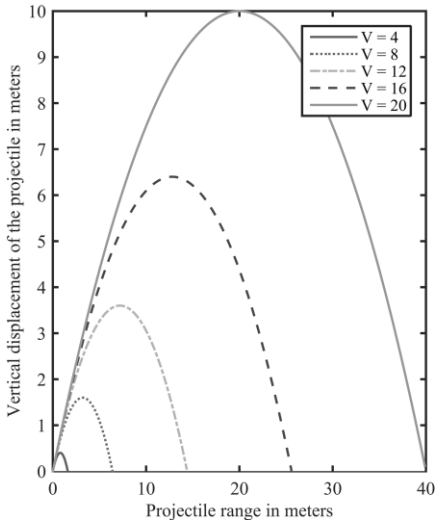

Figure 4. The projectile range with respect to the variation of the projectile initial velocity with an angle equal to 45°and the gravitational acceleration of 10 m/s^2

Now, after performing the previous steps, we can create new members in the algorithm; in other words, the projectiles can be launched. Obviously, each member is recognized by its coordinates in problem space, which are the dimensions' values of each member here. In order to specify the new members, we use the projectiles coordinates and their range amount ($R(x)$) and the dimensions that should be changed in each member. In this way, the coordinates of the new members are determined randomly by the normal probability distribution with the mean value $x_{i,d,}$ and the standard deviation $R(x_{i,d})$, where $x_{i,d}$ represents the mean value of dimension d for member i. Also, the number of new members of the algorithm is the same as the number of initial population members here and in each iteration of the algorithm. As mentioned earlier, the effect of the disruptive forces are applied in two stages due to the experimental observations on the PRO algorithm, the first stage was at the projectiles range $R(x)$ determination time that was in the form of multiplying by random numbers with uniform distribution, and the second stage is implicitly in the new members' creation. As mentioned, in the section of creating new members, we used random numbers with normal distribution for creating new members. These random numbers are implicitly the second phase of applying the disruptive and friction forces. Otherwise, the randomly creation new members is not necessary, and the coordinates of the new member can be obtained by having the current coordinates and range of each member.

Now, in this step of the algorithm, the number of population members in the algorithm has doubled since the beginning of the algorithm main loop reached *2N*; now, it is the time to apply the specified cost function to the new members of the population. This is necessary to perform the next stage, which is sorting all members according to their costs. After applying the cost function on the new members and sorting all the new and old members with respect to their costs, half of the population members with inappropriate (higher) costs are removed from the population. This action is called competitive exclusion.

At the end of the algorithm main loop, the cost of the best member in each iteration can be stored in a variable such as C_{best}. Then the counter of the algorithm loop is increased by one, and we return to the beginning of the loop that is the investigation of the loop end condition.

All of the stages that have mentioned from the beginning of this section are repeated until the end condition of the loop will be satisfied; for example, until the value of the loop repetition counter reaches the predetermined amount. At this point, the repetition ends, and the algorithm is completed. At the end, the algorithm converges to the object point with the maximum fitness and minimum cost.

4. ALGORITHM COMPARISON AND RESULTS

In the PRO algorithm, we aimed for a better performance than other known algorithms in the optimization field. Thus, we compared it with the powerful and well-known particle swarm and differential evolution algorithms. In Tables 2 and 3, the values corresponding to the parameters of these algorithms can be seen. These values are experimental and represent the algorithm's best performance.

In this comparison, we used the test functions that have been introduced in [70] for measuring the algorithms' performance in the IEEE CEC 2017 conference. In addition, we compared the PRO algorithm with the particle swarm and differential evolution algorithms in two applied cases of multi-layer perceptron (MLP) neural networks training and pattern recognition by the GMM method. The obtained results from different cases are presented in Tables 4 to 7. Also, the obtained results for the practical cases have been presented in a graph form in Figures 5 and 6. As can be seen, the results show that the proposed algorithm performs better than the other algorithms.

4. 1. Comparison Using the Test Functions As mentioned earlier, the PRO algorithm was compared with the particle swarm and differential evolution algorithm on some test functions that have been used in the IEEE CEC 2017 conference for algorithm comparison. In [70], the test functions have been introduced comprehensively.

TABLE 2. The parameters and their determined values in the PSO algorithm which are the same and constant for all comparisons

Symbol	Quantity	Value
Iteration	Number of iterations	10E+4
Dimension (D)	Number of problem dimensions	10, 50
MaxFE	Maximum function evaluation	10E+4×D
Particle	Number of particles	D
$\varphi 1$	Value of $\varphi 1$	0.3
$\varphi 2$	Value of $\varphi 2$	0.7
W_{max}	Initial value of inertia weight	0.9
W_{min}	final value of inertia weight	0.4

TABLE 3. The parameters and their determined values in the DE algorithm which are the same and constant for all comparisons

Symbol	Quantity	Value
Iteration	Number of iterations	10E+4
Dimension (D)	Number of problem dimensions	10, 50
MaxFE	Maximum function evaluation	10E+4 × D
PopSize	Number of populations	D
F	Impact factor of difference	1
CR	Probability of crossover	0.5

The results of this comparison are presented in Tables 2, 3, and 4. These comparisons were made similar to the presented method in [70], and based on the error between the difference of the optimization algorithms' output and the ideal output as Equation (6). It should be noted that the purpose of optimization in these comparisons was to find the minimum value of the functions in the given range [-100,100] for each function (minimization).

$$E = C(x_i) - C^*(x_i) \quad (6)$$

where $C(x_i)$ is the output of the optimization algorithm or the minimum acquired cost, and $C^*(x_i)$ is the ideal output or the minimum accessible cost for each function by the optimization algorithms.

Here, the comparisons were performed in such a way that appropriate initial conditions were considered for each of the three algorithms, and these conditions were constant for all comparisons. The initial conditions for the PRO algorithm are presented in Table 1.

The comparison between the optimization algorithms was performed with an equal population size for algorithms. The number of members was equal to the number of problem dimensions (10 and 50). Moreover, each comparison was repeated 50 times, and each time the seed of random number generator was identical for all algorithms. So, the generation of random values would be the same for all algorithms, and the randomly generated numbers for the algorithms would be the same if possible. Also, the seed of the random number generators was different for each order of the comparison repetition. Furthermore, the maximum amount of the function evaluation for all algorithms was assumed to be equal to 10E+4 × D, which was the end condition for the main loop of the algorithms.

As can be seen in the comparison tables of the algorithms (Tables 4, 5, and 6), the results show that the performance of the PRO algorithm generally is much better than other algorithms for all the cases and on all the test functions.

The tables give the best, worst, mean, median, and standard deviation (SD) of the results in 50 complete and separate runs of each algorithm for each test, respectively, in 10 and 50-dimensional states in the corresponding columns. In these tables, the best results in each experiment are written in bold. As can be seen, in most cases the best results are for the PRO algorithm, especially when the dimensions are large.

4. 2. Comparison Using Case Studies In this section, we attempted to evaluate the PRO algorithm in real and practical applications. For this purpose, we compared it

TABLE 4. Comparison results of algorithms using the test functions 1 to 10

Test	Method	Dimensions	Result				
			Best	Worst	Mean	Median	SD
F1	PSO	10	1.24E+06	6.65E+09	1.29E+09	1.14E+09	1.56E+09
		50	1.24E+10	5.83E+10	3.20E+10	3.21E+10	1.06E+10
	PRO	10	1.74E+00	**1.11E+04**	**2.61E+03**	1.56E+03	**2.80E+03**
		50	**2.20E+03**	**3.49E+04**	**1.10E+04**	**8.62E+03**	**7.81E+03**
	DE	10	**7.82E-11**	7.01E+07	2.83E+06	**8.19E-09**	1.39E+07
		50	2.12E+10	3.17E+10	2.75E+10	2.76E+10	2.33E+09
F2	PSO	10	4.03E+04	1.80E+13	1.01E+12	5.77E+08	3.27E+12
		50	1.49E+50	1.16E+74	2.78E+72	3.05E+62	1.65E+73
	PRO	10	6.30E-06	**9.19E-03**	**1.44E-03**	**3.92E-04**	**2.26E-03**
		50	**9.82E-06**	**6.34E+04**	**1.43E+03**	**8.05E+01**	**8.96E+03**
	DE	10	**1.24E-07**	3.03E+08	1.26E+07	2.46E+01	5.99E+07
		50	6.48E+57	2.18E+64	5.27E+62	4.29E+60	3.08E+63
F3	PSO	10	4.35E+03	1.47E+05	3.33E+04	2.06E+04	3.13E+04
		50	1.85E+05	7.84E+05	3.93E+05	3.79E+05	1.26E+05
	PRO	10	6.05E-05	**5.90E-03**	**1.28E-03**	9.70E-04	**1.12E-03**
		50	**1.01E+00**	**1.65E+01**	**4.09E+00**	**3.36E+00**	**2.94E+00**
	DE	10	**1.95E-09**	9.55E+03	4.17E+02	**1.83E-05**	1.52E+03
		50	2.51E+05	3.78E+05	3.15E+05	3.14E+05	2.66E+04
F4	PSO	10	9.29E+00	1.34E+03	1.37E+02	9.57E+01	1.91E+02
		50	1.41E+03	9.93E+03	4.66E+03	3.92E+03	2.32E+03
	PRO	10	**2.09E-03**	7.18E+01	3.60E+00	**9.31E-01**	1.36E+01
		50	**2.10E+01**	**2.18E+02**	**1.14E+02**	**1.08E+02**	**5.64E+01**
	DE	10	1.13E-02	**8.30E+00**	**1.61E+00**	1.20E+00	**1.85E+00**
		50	1.95E+03	3.32E+03	2.55E+03	2.53E+03	3.18E+02
F5	PSO	10	2.46E+01	1.35E+02	7.14E+01	6.89E+01	2.50E+01
		50	3.53E+02	6.75E+02	4.95E+02	4.95E+02	7.58E+01
	PRO	10	6.96E+00	3.38E+01	2.02E+01	1.99E+01	7.22E+00
		50	**1.99E+02**	**5.07E+02**	**3.29E+02**	**3.31E+02**	5.77E+01
	DE	10	**6.14E+00**	**2.05E+01**	**1.41E+01**	**1.46E+01**	**3.00E+00**
		50	5.50E+02	6.39E+02	5.98E+02	5.96E+02	**1.87E+01**
F6	PSO	10	6.45E+00	7.70E+01	4.13E+01	3.96E+01	1.61E+01
		50	5.25E+01	1.06E+02	7.33E+01	7.29E+01	1.03E+01
	PRO	10	**9.71E-05**	**2.54E-03**	**4.23E-04**	**3.00E-04**	**3.97E-04**
		50	**1.04E-02**	**6.53E-01**	**9.77E-02**	**1.64E-02**	**1.74E-01**
	DE	10	1.95E-02	3.91E+00	5.36E-01	3.86E-01	6.58E-01
		50	5.49E+01	6.79E+01	6.15E+01	6.16E+01	2.83E+00
F7	PSO	10	3.54E+01	2.52E+02	9.16E+01	8.44E+01	3.98E+01
		50	7.23E+02	1.78E+03	1.23E+03	1.22E+03	2.40E+02
	PRO	10	**1.09E+01**	4.43E+01	**2.53E+01**	**2.47E+01**	7.20E+00
		50	**2.10E+02**	**3.92E+02**	**2.97E+02**	**2.94E+02**	**4.32E+01**
	DE	10	1.84E+01	**4.24E+01**	3.09E+01	3.10E+01	**4.67E+00**
		50	1.34E+03	1.86E+03	1.67E+03	1.69E+03	1.08E+02

F8	PSO	10	1.76E+01	1.33E+02	4.69E+01	4.46E+01	2.18E+01
		50	3.85E+02	6.77E+02	5.08E+02	5.00E+02	6.67E+01
	PRO	10	**5.97E+00**	5.17E+01	2.27E+01	2.09E+01	**1.05E+01**
		50	**2.28E+02**	**4.80E+02**	**3.39E+02**	3.31E+02	5.61E+01
	DE	10	9.98E+00	**2.40E+01**	**1.68E+01**	**1.70E+01**	3.22E+00
		50	5.63E+02	6.34E+02	5.99E+02	6.01E+02	**1.74E+01**
F9	PSO	10	6.59E+01	4.44E+03	1.02E+03	8.61E+02	7.72E+02
		50	1.17E+04	3.46E+04	2.11E+04	2.09E+04	4.98E+03
	PRO	10	**8.81E-08**	1.11E+03	4.36E+01	**5.08E-06**	1.84E+02
		50	**6.27E+03**	**2.24E+04**	**1.24E+04**	**1.31E+04**	3.42E+03
	DE	10	1.32E-05	**7.70E-01**	**7.76E-02**	2.68E-02	**1.37E-01**
		50	2.10E+04	3.60E+04	2.88E+04	2.84E+04	**3.20E+03**
F10	PSO	10	6.49E+02	2.54E+03	1.47E+03	1.46E+03	4.33E+02
		50	6.34E+03	1.11E+04	9.37E+03	9.40E+03	1.01E+03
	PRO	10	**1.25E+02**	1.07E+03	6.08E+02	6.03E+02	2.22E+02
		50	**3.39E+03**	**6.96E+03**	**5.38E+03**	**5.45E+03**	8.44E+02
	DE	10	2.01E+02	**8.53E+02**	**5.43E+02**	**5.37E+02**	**1.42E+02**
		50	1.15E+04	1.30E+04	1.25E+04	1.25E+04	**3.45E+02**

TABLE 5. Comparison results of algorithms using the test functions 11 to 20

Test	**Method**	**Dimensions**	**Result**				
			Best	**Worst**	**Mean**	**Median**	**SD**
F11	PSO	10	1.44E+01	7.26E+03	9.47E+02	3.38E+02	1.53E+03
		50	1.83E+03	3.14E+04	9.15E+03	7.56E+03	6.81E+03
	PRO	10	1.02E+00	**2.41E+01**	**1.09E+01**	1.10E+01	**5.75E+00**
		50	**1.05E+02**	**3.52E+02**	**2.23E+02**	**2.27E+02**	**5.23E+01**
	DE	10	**4.10E-01**	3.11E+02	2.21E+01	**7.75E+00**	4.86E+01
		50	2.03E+03	3.88E+03	2.59E+03	2.58E+03	3.38E+02
F12	PSO	10	2.41E+04	5.55E+08	4.29E+07	3.36E+06	1.21E+08
		50	1.76E+09	3.94E+10	8.98E+09	7.07E+09	6.80E+09
	PRO	10	**4.77E+02**	**4.18E+04**	**1.39E+04**	1.04E+04	**1.28E+04**
		50	**2.29E+05**	**4.76E+06**	**2.04E+06**	**1.77E+06**	**1.16E+06**
	DE	10	8.48E+02	4.94E+06	1.99E+05	**1.61E+03**	9.78E+05
		50	1.01E+09	1.96E+09	1.42E+09	1.42E+09	1.97E+08
F13	PSO	10	1.03E+03	8.64E+04	1.84E+04	1.33E+04	1.86E+04
		50	1.80E+05	1.27E+10	2.09E+09	1.41E+09	2.61E+09
	PRO	10	2.25E+01	3.13E+04	1.22E+04	9.60E+03	1.11E+04
		50	**2.18E+02**	**3.66E+04**	**9.42E+03**	**4.09E+03**	**1.06E+04**
	DE	10	**6.40E+00**	**1.34E+03**	**1.36E+02**	**6.89E+01**	**2.47E+02**
		50	1.83E+05	1.77E+06	6.69E+05	5.92E+05	3.50E+05
F14	PSO	10	1.57E+02	2.74E+04	8.31E+03	3.19E+03	8.51E+03
		50	1.24E+05	3.19E+07	3.44E+06	1.39E+06	5.44E+06
	PRO	10	6.28E+00	2.26E+04	5.96E+03	2.77E+03	6.85E+03
		50	**1.68E+04**	**3.79E+05**	**9.05E+04**	**7.84E+04**	**6.88E+04**
	DE	10	**2.93E-01**	**3.24E+01**	**2.00E+01**	**2.13E+01**	**7.38E+00**
		50	2.36E+05	1.18E+06	5.49E+05	5.29E+05	1.97E+05

F15	PSO	10	2.17E+02	1.09E+05	2.81E+04	2.01E+04	2.65E+04
		50	3.27E+04	4.19E+08	1.14E+07	1.16E+05	6.05E+07
	PRO	10	2.21E+00	2.57E+04	6.74E+03	4.55E+03	6.68E+03
		50	**2.30E+02**	**3.08E+04**	**8.91E+03**	**8.30E+03**	**7.50E+03**
	DE	10	**1.09E-01**	**1.26E+02**	**3.20E+01**	**2.06E+01**	**3.29E+01**
		50	1.42E+04	5.94E+04	3.45E+04	3.26E+04	1.27E+04
F16	PSO	10	1.87E+02	8.83E+02	4.90E+02	4.96E+02	1.54E+02
		50	1.30E+03	4.34E+03	2.76E+03	2.72E+03	6.92E+02
	PRO	10	**5.56E-01**	4.99E+02	2.21E+02	2.19E+02	1.39E+02
		50	**9.10E+02**	**2.93E+03**	**1.92E+03**	**1.90E+03**	3.86E+02
	DE	10	2.78E+00	**3.10E+02**	**8.65E+01**	**4.39E+01**	**7.64E+01**
		50	2.70E+03	3.76E+03	3.29E+03	3.30E+03	**2.47E+02**
F17	PSO	10	4.19E+01	6.11E+02	1.85E+02	1.26E+02	1.23E+02
		50	1.50E+03	1.15E+04	2.48E+03	2.12E+03	1.40E+03
	PRO	10	**1.31E+00**	1.78E+02	5.17E+01	3.17E+01	4.88E+01
		50	**7.07E+02**	2.30E+03	**1.44E+03**	**1.38E+03**	3.27E+02
	DE	10	8.55E+00	**1.20E+02**	**3.29E+01**	**2.45E+01**	**2.02E+01**
		50	1.51E+03	**2.20E+03**	1.88E+03	1.89E+03	**1.67E+02**
F18	PSO	10	7.79E+02	5.39E+04	1.80E+04	1.13E+04	1.68E+04
		50	8.83E+05	1.51E+08	1.44E+07	7.00E+06	2.50E+07
	PRO	10	1.99E+02	3.69E+04	1.09E+04	7.17E+03	9.88E+03
		50	**5.13E+04**	**1.19E+06**	**5.37E+05**	**5.48E+05**	**2.75E+05**
	DE	10	**2.01E+01**	**2.71E+03**	**1.51E+02**	**3.84E+01**	**5.20E+02**
		50	2.11E+06	1.23E+07	6.30E+06	6.27E+06	2.18E+06
F19	PSO	10	1.54E+02	1.99E+07	4.31E+05	1.33E+04	2.80E+06
		50	3.57E+05	1.38E+09	3.69E+07	4.60E+06	1.95E+08
	PRO	10	3.58E+00	2.81E+04	8.08E+03	5.46E+03	8.47E+03
		50	**1.59E+02**	**4.25E+04**	**1.65E+04**	**1.55E+04**	**1.21E+04**
	DE	10	**3.43E-01**	**2.19E+02**	**1.28E+01**	**2.92E+00**	**3.49E+01**
		50	6.15E+02	1.71E+05	4.75E+04	3.77E+04	3.95E+04
F20	PSO	10	9.11E+01	5.14E+02	2.48E+02	2.44E+02	1.01E+02
		50	9.31E+02	2.54E+03	1.66E+03	1.65E+03	3.66E+02
	PRO	10	**2.74E-03**	**3.42E+01**	**1.17E+01**	**6.78E+00**	**1.09E+01**
		50	**5.07E+02**	1.85E+03	**1.15E+03**	**1.14E+03**	2.92E+02
	DE	10	4.17E-02	2.07E+02	2.92E+01	2.12E+01	3.44E+01
		50	9.69E+02	**1.68E+03**	1.37E+03	1.43E+03	**1.79E+02**

TABLE 6. Comparison results of algorithms using the test functions 21 to 30

Test	Method	Dimensions	Result				
			Best	Worst	Mean	Median	SD
F21	PSO	10	1.15E+02	3.30E+02	2.46E+02	2.60E+02	5.60E+01
		50	5.66E+02	9.96E+02	7.45E+02	7.51E+02	9.93E+01
	PRO	10	**1.00E+02**	2.69E+02	2.22E+02	2.35E+02	**5.14E+01**
		50	**4.07E+02**	**6.96E+02**	**5.48E+02**	**5.37E+02**	5.65E+01
	DE	10	1.02E+02	**2.34E+02**	**1.74E+02**	**2.20E+02**	6.06E+01
		50	7.08E+02	8.14E+02	7.82E+02	7.88E+02	**2.12E+01**

F22	PSO	10	1.05E+02	2.73E+03	5.22E+02	1.79E+02	6.38E+02
		50	3.53E+03	1.12E+04	9.43E+03	9.81E+03	1.34E+03
	PRO	10	1.00E+02	1.60E+03	3.58E+02	**1.04E+02**	4.73E+02
		50	**1.00E+02**	**8.26E+03**	**6.09E+03**	**6.26E+03**	1.49E+03
	DE	10	**1.43E+01**	**1.53E+02**	**9.64E+01**	1.06E+02	**3.35E+01**
		50	1.14E+04	1.36E+04	1.28E+04	1.29E+04	**3.91E+02**
F23	PSO	10	3.39E+02	6.49E+02	4.13E+02	3.91E+02	7.37E+01
		50	9.69E+02	1.80E+03	1.43E+03	1.41E+03	1.98E+02
	PRO	10	3.13E+02	3.71E+02	3.36E+02	3.34E+02	1.51E+01
		50	**7.02E+02**	**9.98E+02**	**8.33E+02**	**8.35E+02**	7.72E+01
	DE	10	**3.11E+02**	**3.30E+02**	**3.18E+02**	**3.18E+02**	**4.50E+00**
		50	9.09E+02	1.03E+03	9.96E+02	9.97E+02	**2.30E+01**
F24	PSO	10	1.17E+02	5.96E+02	3.97E+02	4.15E+02	1.16E+02
		50	1.07E+03	1.81E+03	1.35E+03	1.34E+03	1.60E+02
	PRO	10	**1.00E+02**	4.64E+02	3.67E+02	3.89E+02	9.37E+01
		50	**8.86E+02**	1.39E+03	1.07E+03	1.04E+03	1.18E+02
	DE	10	1.17E+02	**3.71E+02**	**3.18E+02**	**3.54E+02**	**7.49E+01**
		50	9.47E+02	**1.05E+03**	**1.01E+03**	**1.02E+03**	**2.12E+01**
F25	PSO	10	4.10E+02	1.21E+03	5.17E+02	4.74E+02	1.42E+02
		50	1.21E+03	6.76E+03	3.13E+03	2.70E+03	1.45E+03
	PRO	10	2.00E+02	5.24E+02	4.30E+02	4.45E+02	**4.20E+01**
		50	**4.61E+02**	**5.81E+02**	**5.35E+02**	**5.40E+02**	**3.64E+01**
	DE	10	**1.16E+02**	**4.33E+02**	**3.90E+02**	**3.99E+02**	5.34E+01
		50	2.00E+03	3.45E+03	2.78E+03	2.81E+03	3.48E+02
F26	PSO	10	3.19E+02	2.09E+03	1.27E+03	1.40E+03	5.39E+02
		50	5.81E+03	1.41E+04	9.71E+03	9.67E+03	1.63E+03
	PRO	10	**2.00E+02**	1.76E+03	8.87E+02	7.08E+02	5.52E+02
		50	**3.45E+03**	**6.45E+03**	**4.93E+03**	**4.90E+03**	6.84E+02
	DE	10	3.00E+02	**4.30E+02**	**3.77E+02**	**3.87E+02**	**3.92E+01**
		50	5.55E+03	7.35E+03	6.93E+03	6.98E+03	**3.05E+02**
F27	PSO	10	4.08E+02	5.84E+02	4.77E+02	4.80E+02	4.48E+01
		50	9.92E+02	2.09E+03	1.34E+03	1.29E+03	2.35E+02
	PRO	10	**3.89E+02**	5.01E+02	4.25E+02	4.09E+02	3.38E+01
		50	6.48E+02	1.13E+03	8.59E+02	8.65E+02	1.23E+02
	DE	10	3.90E+02	**4.10E+02**	**3.95E+02**	**3.95E+02**	**3.85E+00**
		50	**6.13E+02**	**7.49E+02**	**6.82E+02**	**6.80E+02**	**2.91E+01**
F28	PSO	10	3.16E+02	1.11E+03	6.53E+02	6.19E+02	1.79E+02
		50	1.44E+03	7.81E+03	3.96E+03	3.69E+03	1.61E+03
	PRO	10	**3.00E+02**	6.46E+02	5.01E+02	5.84E+02	1.39E+02
		50	**4.59E+02**	**6.10E+02**	**4.98E+02**	**5.04E+02**	**2.62E+01**
	DE	10	3.70E+02	**6.12E+02**	**4.46E+02**	**4.29E+02**	**6.42E+01**
		50	1.16E+03	4.30E+03	2.57E+03	2.43E+03	8.09E+02
F29	PSO	10	3.13E+02	9.18E+02	5.46E+02	5.21E+02	1.54E+02
		50	2.54E+03	7.62E+03	3.73E+03	3.63E+03	9.54E+02
	PRO	10	2.52E+02	5.00E+02	3.30E+02	3.23E+02	5.98E+01
		50	**5.31E+02**	**2.23E+03**	**1.32E+03**	**1.31E+03**	3.59E+02
	DE	10	**2.38E+02**	**2.86E+02**	**2.54E+02**	**2.53E+02**	**1.13E+01**
		50	2.09E+03	3.24E+03	2.78E+03	2.80E+03	**2.76E+02**

F30	PSO	10	1.00E+04	1.56E+07	2.84E+06	1.87E+06	3.23E+06
		50	4.72E+07	2.75E+09	3.76E+08	1.44E+08	5.52E+08
	PRO	10	1.43E+03	1.28E+06	**2.67E+05**	**1.63E+04**	4.39E+05
		50	**6.55E+05**	**2.25E+06**	**1.11E+06**	**1.02E+06**	**3.63E+05**
	DE	10	**1.34E+03**	**8.83E+05**	3.67E+05	1.71E+05	**3.70E+05**
		50	1.74E+07	4.98E+07	3.10E+07	2.98E+07	8.71E+06

with the PSO and DE algorithms in two practical cases, one is relevant to minimizing the mean squared error (MSE) in training level of the MLP neural networks, and the other is to obtain the most suitable pattern for a predetermined Gaussian mixture by the Gaussian mixture model (GMM) method. Both cases can be considered as a type of minimization. As can be seen in Table 7 and Figures 5 and 6, the performance of the PRO algorithm in these cases is better than other algorithms.

In the case that was relevant to the MLP neural network training [71], the MLP method was used for the classification of the Iris flower dataset. In this case, the purpose was to minimize the cost function, mean squared error, in the MLP neural network training. This error shows the difference between the actual and ideal output of the neural network. The number of hidden neurons was considered equal to 10, in which case the dimensions of the vector of neural network parameters (weights and biases) had 83 dimensions for optimization. Moreover, 50% of the whole dataset was used for training the neural network, in other words, we set the value of the variable, fraction of data for training, equal to 0.5.

In the application about the GMM, it should be noted that this model has several applications in different problems such as data processing, for example, speech and image processing, as in [72, 73]. The application of GMM is usually in the pattern recognition and modeling stage. In a standard approach for the training of the Gaussian mixture model parameters, the EM algorithm [74] has been used; however, we used the PSO, DE, and PRO algorithms for comparison instead of EM algorithm. In this case, we used a Gaussian test model with four Gaussian components by ten dimensions. Then by using GMM that was based on the intended optimization algorithms, the best pattern for this model was adapted. In this experiment, based on the determined properties for the test model, the number of the dimensions of the GMM vector parameters for the optimization was equal to 84, and these parameters included mean vectors, covariance matrix, and Gaussian weights.

Figures 5 and 6 show that the PSO and DE algorithms have the drawback of early convergence to the local optimum points and stick in these local optimum points. According to these graphs, PSO and DE algorithms have been converged approximately in iteration 200 and even earlier. However, the PRO algorithm performed much better than these two algorithms from this point of view. As can be seen from these figures, the PRO algorithm has not been fully converged even until the last iteration of the algorithm. It can be drawn from results that the PRO did not stick in local optimum points and tended to more optimization. In addition, the speed of convergence to the global optimum point was much higher in the PRO algorithm than the other two algorithms in comparison. this notable difference in the convergence speed of algorithms can be clearly seen in Figures 5 and 6. Also in Table 7, the best results in each test are written in bold. As can be seen, in both cases the best results are for the PRO algorithm.

TABLE 7. algorithms comparison results in GMM and MLP experiment in 50 separate runs and population members quantity equal to 10 members and the maximum function evaluation of 10E+4

Test	Method	Result				
		Best	**Worst**	**Mean**	**Median**	**SD**
MLP	PSO	1.13E-01	1.02E+00	2.93E-01	2.59E-01	1.52E-01
	PRO	**5.12E-03**	**1.86E-01**	**5.75E-02**	**5.06E-02**	**3.74E-02**
	DE	6.62E-01	2.01E+00	1.29E+00	1.33E+00	2.81E-01
GMM	PSO	5.16E+01	5.61E+01	5.37E+01	5.38E+01	1.17E+00
	PRO	**3.74E+01**	**5.29E+01**	**4.25E+01**	**4.13E+01**	3.56E+00
	DE	5.41E+01	5.58E+01	5.49E+01	5.49E+01	**3.64E-01**

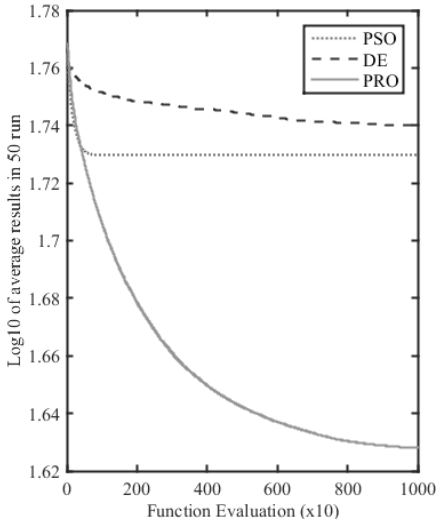

Figure 5. Comparison of the convergence rate of the algorithms in the GMM test

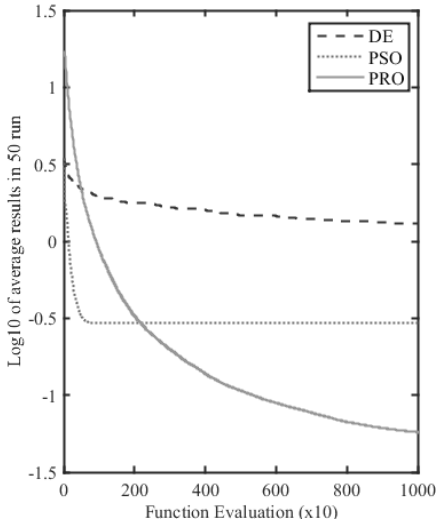

Figure 6. Comparison of the convergence rate of the algorithms in the MLP test

5. CONCLUSION

In this paper, we introduced a new optimization algorithm. The projectiles optimization (PRO) algorithm is a new metaheuristic optimization algorithm that its method is based on the projectiles' motion, and its main idea is based on the properties of this type of motion in physics. In addition, the PRO algorithm is a population-based algorithm.

In this essay, we attempted to introduce the PRO algorithm and analyze its performance and compare it with the two most powerful algorithms in the optimization field, in which they were selected among the metaheuristic optimization algorithms and were most similar to the PRO algorithm. These comparisons were performed to evaluate the accuracy and performance level of the PRO algorithm with respect to other algorithms.

The obtained results from the comparison of PRO algorithm with PSO and DE algorithms generally showed the preference of this algorithm in all the cases and even despite the limitations applied on the comparison to get more fair conditions. The preference of the performance and accuracy of the PRO algorithm was much more notable than the PSO and DE algorithms, especially in high dimensional tests.

In the experiments, we observed that the PRO algorithm had a higher degree of desire for optimization than other algorithms. This means that the PRO algorithm performs much better than the other algorithms and has not the early convergence and sticking in local optimum points, which are considered as drawbacks and weak points. Furthermore, the speed of convergence to the global optimum point in the PRO algorithm is much higher than other algorithms. In other words, the PRO algorithm is able to provide a better balance between the exploration and exploitation compared to the other algorithms. Because a metaheuristic algorithm performs more efficiently if it can provide an appropriate balance between diversification and intensification.

In the next work, we intend to improve the power and accuracy of the PRO algorithm to the best possible level by using the dynamic and optimum updating and self-adaptive initialization of variables associated with the algorithm.

6. REFERENCES

1. Talbi, E.G., "Metaheuristics: From design to implementation",Vol. 74, John Wiley & Sons. (2009). https://doi.org/10.1002/9780470496916
2. Birattari, M., Paquete, L., Strutzle, T. and Varrentrapp, K., Classification of metaheuristics and design of experiments for the analysis of components. Technical Report No. AIDA-01-05, (2001). Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.12.4407&rep=rep1&type=pdf
3. Boussaïd, I., Lepagnot, J. and Siarry, P., "A survey on optimization metaheuristics," ***Information Sciences***, Vol. 237, (2013), 82-117. https://doi.org/10.1016/j.ins.2013.02.041
4. Blum, C. and Roli, A., "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," ***ACM Computing Surveys (CSUR)***, Vol. 35, No. 3, (2003), 268-308. https://doi.org/10.1145/937503.937505
5. Bianchi, L., Dorigo, M., Gambardella, L.M. and Gutjahr, W.J., "A survey on metaheuristics for stochastic combinatorial optimization," ***Natural Computing: an International Journal***, Vol. 8, No. 2, (2009), 239-287. https://doi.org/10.1007/s11047-008-9098-4
6. Goldberg, D.E. and Deb, K., "A comparative analysis of selection schemes used in genetic algorithms," ***Foundations of genetic***

algorithms, Vol. 1, (1991), 69-93. https://doi.org/10.1016/b978-0-08-050684-5.50008-2

7. Blickle, T. and Thiele, L., A comparison of selection schemes used in genetic algorithms. (1995), TIK-Report. https://doi.org/10.1162/evco.1996.4.4.361
8. Beasley, D., Bull, D.R. and Martin, R.R., "An overview of genetic algorithms: Part 2, research topics," ***University computing***, Vol. 15, No. 4, (1993), 170-181. Retrieved from http://citeseer.ist.psu.edu/16527.html
9. Becerra, R.L. and Coello, C.A.C., A cultural algorithm with differential evolution to solve constrained optimization problems, in Advances in Artificial Intelligence–IBERAMIA 2004. Springer, (20040, 881-890. https://doi.org/10.1007/978-3-540-30498-2_88
10. Konak, A., Coit, D.W. and Smith, A.E., "Multi-objective optimization using genetic algorithms: A tutorial," ***Reliability Engineering & System Safety***, Vol. 91, No. 9, (2006), 992-1007. https://doi.org/10.1016/j.ress.2005.11.018
11. Alba, E. and Troya, J.M., "A survey of parallel distributed genetic algorithms," ***Complexity***, Vol. 4, No. 4, (1999), 31-52. https://doi.org/10.1002/(SICI)1099-0526(199903/04)4:4<31::AID-CPLX5>3.0.CO;2-4
12. Elsayed, S.M., Sarker, R.A. and Essam, D.L., "A new genetic algorithm for solving optimization problems," ***Engineering Applications of Artificial Intelligence***, Vol. 27, (2014), 57-69. https://doi.org/10.1016/j.engappai.2013.09.013
13. Goldberg, D.E., "Genetic algorithms in search optimization and machine learning," Addison-Wesley Reading Menlo Park, Vol. 412, (1989). https://doi.org/10.5860/choice.27-0936
14. Kirkpatrick, S. and Vecchi, M.P., "Optimization by simulated annealing," ***Science***, Vol. 220, No. 4598, (1983), 671-680. https://doi.org/10.1126/science.220.4598.671
15. Černý, V., "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm," ***Journal of Optimization Theory and Applications***, Vol. 45, No. 1, (1985), 41-51. https://doi.org/10.1007/BF00940812
16. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H. and Teller, E., "Equation of state calculations by fast computing machines," ***The Journal of Chemical Physics***, Vol. 21, No. 6, (1953), 1087-1092. https://doi.org/10.1063/1.1699114
17. Creutz, M., "Microcanonical Monte Carlo simulation," ***Physical Review Letters***, Vol. 50, No. 19, (1983), 1411-1414. https://doi.org/10.1103/PhysRevLett.50.1411
18. Dueck, G. and Scheuer, T., "Threshold accepting: A general-purpose optimization algorithm appearing superior to simulated annealing," ***Journal of Computational Physics***, Vol. 90, No. 1, (1990), 161-175. https://doi.org/10.1016/0021-9991(90)90201-B
19. Dréo, J., Petrowski, A., Siarry, P. and Taillard, E., "Metaheuristics for hard optimization: Methods and case studies, Springer Science & Business Media, (2006).
20. Charon, I. and Hudry, O., "The noising method: A new method for combinatorial optimization," ***Operations Research Letters***, Vol. 14, No. 3, (1993), 133-137. https://doi.org/10.1016/0167-6377(93)90023-A
21. Charon, I. and Hudry, O., "The noising methods: A generalization of some metaheuristics," ***European Journal of Operational Research***, Vol. 135, No. 1, (2001), 86-101. https://doi.org/10.1016/S0377-2217(00)00305-2
22. Charon, I. and Hudry, O., The noising methods: A survey, in Essays and surveys in metaheuristics. 2002, Springer.245-261. https://doi.org/10.1007/978-1-4615-1507-4
23. Charon, I. and Hudry, O., "Self-tuning of the noising methods," ***Optimization***, Vol. 58, No. 7, (2009), 823-843. https://doi.org/10.1080/02331930902944911
24. Courat, J.-P., Raynaud, G., Mrad, I. and Siarry, P., "Electronic component model minimization based on log simulated annealing," ***IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications***, Vol. 41, No. 12, (1994), 790-795. https://doi.org/10.1109/81.340841
25. Jeong, I.-K. and Lee, J.-J., "Adaptive simulated annealing genetic algorithm for system identification," ***Engineering Applications of Artificial Intelligence***, Vol. 9, No. 5, (1996), 523-532. https://doi.org/10.1016/0952-1976(96)00049-8
26. Suman, B. and Kumar, P., "A survey of simulated annealing as a tool for single and multiobjective optimization," ***Journal of the Operational Research Society***, Vol. 57, No. 10, (2006), 1143-1160. https://doi.org/10.1057/palgrave.jors.2602068
27. Chopard, B., and Tomassini, M., "Simulated annealing, Natural Computing Series (First Edition). IN-TECH Education and Publishing, (2018). https://doi.org/10.1007/978-3-319-93073-2_4
28. Hasani, A. and Soltani, R., "A hybrid meta-heuristic for the dynamic layout problem with transportation system design," ***International Journal of Engineering - Transactions B: Applications,*** Vol. 28, No. 8, (2015), 1175-1185. https://doi.org/10.5829/idosi.ije.2015.28.08b.10
29. Fallah, M., Mohajeri, A. and Barzegar-Mohammadi, M., "A new mathematical model to optimize a green gas network: A case study," In CIE 2016: 46th International Conferences on Computers and Industrial Engineering, 1142–1149. Retrieved from https://www.researchgate.net/publication/320920905
30. Chan, F.T. and Tiwari, M.K., "Swarm Intelligence: Focus on Ant and Particle Swarm Optimization, IntechOpen, (2007). https://doi.org/10.5772/5121
31. Saenphon, T., Phimoltares, S. and Lursinsap, C., "Combining new fast opposite gradient search with ant colony optimization for solving travelling salesman problem," ***Engineering Applications of Artificial Intelligence***, Vol. 35, (2014), 324-334. https://doi.org/10.1016/j.engappai.2014.06.026
32. Dorigo, M. and Stutzle, T., The ant colony optimization metaheuristic: Algorithms, applications, and advances, In Handbook of metaheuristics. Springer, (2003), 250-285. https://doi.org/10.7551/mitpress/1290.003.0004
33. Dorigo, M., Birattari, M. and Stutzle, T., "Ant colony optimization," ***IEEE Computational Intelligence Magazine***, Vol. 1, No. 4, (2006), 28-39. https://doi.org/10.1109/MCI.2006.329691
34. Dorigo, M. and Stützle, T., Ant colony optimization: Overview and recent advances, In International Series in Operations Research and Management Science (Vol. 272), Handbook of metaheuristics, Springer, 311–351. https://doi.org/10.1007/978-3-319-91086-4_10
35. Mohajeri, A., Mahdavi, I., Mahdavi-Amiri, N. and Tafazzoli, R., "Optimization of tree-structured gas distribution network using ant colony optimization: A case study," ***International Journal of Engineering - Transactions A: Basics***, Vol. 25, No. 2, (2012), 141-158. https://doi.org/10.5829/idosi.ije.2012.25.02a.04
36. Eberhart, R.C. and Kennedy, J., "A new optimizer using particle swarm theory," In Proceedings of the International Symposium on Micro Machine and Human Science, Vol. 1, IEEE, (1995), 39-43. https://doi.org/10.1109/mhs.1995.494215
37. Kennedy, J., Eberhart, R.C. and Shi, Y., "Swarm Intelligence", IEEE Technology and Society Magazine (Vol. 21). Morgan Kaufmann. IEEE, (2002). https://doi.org/10.1109/MTAS.2002.993595
38. Kennedy, J. and Mendes, R., "Population structure and particle swarm performance," In The 2002 IEEE Congress on Evolutionary

Computation, CEC'02, (2002), 1671-1676. https://doi.org/10.1109/CEC.2002.1004493

39. Gulcu, S. and Kodaz, H., "A novel parallel multi-swarm algorithm based on comprehensive learning particle swarm optimization," ***Engineering Applications of Artificial Intelligence***, Vol. 45, (2015), 33-45. https://doi.org/10.1016/j.engappai.2015.06.013
40. Shi, Y. and Eberhart, R., "A modified particle swarm optimizer," In The 1998 IEEE International Conference on Evolutionary Computation, CEC'98, (1998), 69-73. https://doi.org/10.1109/ICEC.1998.699146
41. Shi, Y. and Eberhart, R.C., "Empirical study of particle swarm optimization," In The 1999 IEEE Congress on Evolutionary Computation , CEC'99, (1999), 1945-1950. https://doi.org/10.1109/CEC.1999.785511
42. Clerc, M. and Kennedy, J., "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," ***IEEE Transactions on Evolutionary Computation,*** Vol. 6, No. 1, (2002), 58-73. https://doi.org/10.1109/4235.985692
43. Ozcan, E. and Mohan, C.K., "Particle swarm optimization: Surfing the waves," in The 1999 IEEE Congress on Evolutionary Computation, CEC'99, (1999), 1939-1944. https://doi.org/10.1109/CEC.1999.785510
44. Van den Bergh, F. and Engelbrecht, A.P., "A study of particle swarm optimization particle trajectories," ***Information Sciences***, Vol. 176, No. 8, (2006), 937-971. https://doi.org/10.1016/j.ins.2005.02.003
45. Kennedy, J. and Eberhart, R.C., "A discrete binary version of the particle swarm algorithm," In The 1997 IEEE International Conference on Systems, Man, and Cybernetics, (1997), 4104-4108. https://doi.org/10.1109/icsmc.1997.637339
46. Blackwell, T., Particle swarm optimization in dynamic environments, In Evolutionary computation in dynamic and uncertain environments. 2007, Springer. 29-49. https://doi.org/10.1007/978-3-540-49774-5_2
47. Jam, S., Shahbahrami, A. and Sojoudi Ziyabari, S., "Parallel implementation of particle swarm optimization variants using graphics processing unit platform," ***International Journal of Engineering - Transactions A: Basics***, Vol. 30, No. 1, (2017), 48-56. https://doi.org/10.5829/idosi.ije.2017.30.01a.07
48. Reyes-Sierra, M. and Coello, C.C., "Multi-objective particle swarm optimizers: A survey of the state-of-the-art," ***International Journal of Computational Intelligence Research***, Vol. 2, No. 3, (2006), 287-308. https://doi.org/10.5019/j.ijcir.2006.68
49. Ling, S.H., Chan, K.Y., Leung, F.H.F., Jiang, F. and Nguyen, H., "Quality and robustness improvement for real-world industrial systems using a fuzzy particle swarm optimization," ***Engineering Applications of Artificial Intelligence***, Vol. 47, (2016), 68-80. https://doi.org/10.1016/j.engappai.2015.03.003
50. Mahmoodabadi, M., Taherkhorsandi, M. and Safikhani, H., "Modeling and hybrid pareto optimization of cyclone separators using group method of data handling (gmdh) and particle swarm optimization (pso)," ***International Journal of Engineering - Transactions C: Aspects***, Vol. 26, No. 9, (2012), 1089-1102. https://doi.org/10.5829/idosi.ije.2013.26.09c.15
51. Valdez, F., Melin, P. and Castillo, O., "An improved evolutionary method with fuzzy logic for combining particle swarm optimization and genetic algorithms," ***Applied Soft Computing***, Vol. 11, No. 2, (2011), 2625-2632. https://doi.org/10.1016/j.asoc.2010.10.010
52. Poli, R., "Analysis of the publications on the applications of particle swarm optimisation," ***Journal of Artificial Evolution and Applications***, Vol. 2008, (2008), 1-10. https://doi.org/10.1155/2008/685175
53. Poli, R., Kennedy, J. and Blackwell, T., "Particle swarm optimization," ***Swarm Intelligence***, Vol. 1, No. 1, (2007), 33-57. https://doi.org/10.1007/s11721-007-0002-0
54. Pant, M., Thangaraj, R. and Abraham, A., Particle swarm optimization: Performance tuning and empirical analysis, In Foundations of computational intelligence. 2009, Springer. 101-128. https://doi.org/10.1007/978-3-642-01085-9_5
55. Thangaraj, R., Pant, M., Abraham, A. and Bouvry, P., "Particle swarm optimization: Hybridization perspectives and experimental illustrations," ***Applied Mathematics and Computation***, Vol. 217, No. 12, (2011), 5208-5226. https://doi.org/10.1016/j.amc.2010.12.053
56. Deepa, S., Babu, S.R. and Ranjani, M., "A robust statcom controller using particle swarm optimization," ***International Journal of Engineering - Transactions B: Applications***, Vol. 27, No. 5, (2013), 731-738. https://doi.org/10.5829/idosi.ije.2014.27.05b.08
57. Daliri, H., Mokhtari, H. and Nakhai, I., "A particle swarm optimization approach to joint location and scheduling decisions in a flexible job shop environment," ***International Journal of Engineering-Transaction C: Aspects***, Vol. 28, No. 12, (2015), 1756-1764. https://doi.org/10.5829/idosi.ije.2015.28.12c.08
58. Storn, R. and Price, K., "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," ***Journal of Global Optimization***, Vol. 11, No. 4, (1997), 341-359. https://doi.org/10.1023/A:1008202821328
59. Mezura-Montes, E., Reyes-Sierra, M. and Coello, C.A.C., Multi-objective optimization using differential evolution: A survey of the state of the art, In Advances in differential evolution. Vol. 143, (2008), Springer. 173-196. https://doi.org/10.1007/978-3-540-68830-3_7
60. Amirian, H. and Sahraeian, R., "Multi-objective differential evolution for the flow shop scheduling problem with a modified learning effect," ***International Journal of Engineering - Transactions C: Aspects***, Vol. 27, No. 9, (2014), 1395-1404. https://doi.org/10.5829/idosi.ije.2014.27.09c.09
61. Angeline, P.J., "Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences," In Evolutionary Programming VII, Springer. (1998), 601-610. https://doi.org/10.1007/bfb0040811
62. Price, K., Storn, R.M. and Lampinen, J.A., "Differential evolution: A practical approach to global optimization, Springer Science & Business Media, (2006).
63. Das, S. and Suganthan, P.N., "Differential evolution: A survey of the state of the art," ***IEEE Transactions on Evolutionary Computation,*** Vol. 15, No. 1, (2011), 4-31. https://doi.org/10.1109/TEVC.2010.2059031
64. Karci, A., Imitation of bee reproduction as a crossover operator in genetic algorithms, in PRICAI 2004: Trends in artificial intelligence. (Vol. 3157), Springer, (2004). 1015-1016. https://doi.org/10.1007/978-3-540-28633-2_141
65. Brest, J. and Maučec, M.S., "Self-adaptive differential evolution algorithm using population size reduction and three strategies," ***Soft Computing***, Vol. 15, No. 11, (2011), 2157-2174. https://doi.org/10.1007/s00500-010-0644-5
66. Teng, N.S., Teo, J. and Hijazi, M.H.A., "Self-adaptive population sizing for a tune-free differential evolution," ***Soft Computing***, Vol. 13, No. 7, (2009), 709-724. https://doi.org/10.1007/s00500-008-0344-6
67. Liu, J. and Lampinen, J., "A fuzzy adaptive differential evolution algorithm," ***Soft Computing***, Vol. 9, No. 6, (2005), 448-462. https://doi.org/10.1007/s00500-004-0363-x
68. Tummala, A.S., Chintala, M.R. and Pilla, R., "Tuning of extended kalman filter using self-adaptive differential evolution algorithm for

sensorless permanent magnet synchronous motor drive," ***International Journal of Engineering - Transactions A: Basics***, Vol. 29, No. 11, (2016), 1565-1573. https://doi.org/10.5829/idosi.ije.2016.29.11b.00

69. Kahrizi, M.R. Projectiles optimization (pro) algorithm. 2017 [cited 2020 Jun. 22]; IEEE Dataport. Available from: https://doi.org/10.21227/H2TK92
70. Awad, N.H., Ali, M.Z., Liang, J.J., Qu, B.Y. and Suganthan, P.N., Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective bound constrained real-parameter numerical optimization. 2016. Available: https://www.ntu.edu.sg/home/EPNSugan/index_files/CEC2017/CEC 2017.htm. Accessed: 21/06/2020.
71. Haykin, S., "Neural networks: A comprehensive foundation, Prentice Hall PTR, (1994).
72. Reynolds, D.A. and Rose, R.C., "Robust text-independent speaker identification using Gaussian mixture speaker models," ***IEEE Transactions on Speech and Audio Processing***, Vol. 3, No. 1, (1995), 72-83. https://doi.org/10.1016/0167-6393(95)00009-D
73. Kahrizi, M.R. and Kabudian, S.J., "Long-term spectral pseudo-entropy (ltspe): A new robust feature for speech activity detection," ***Journal of Information Systems & Telecommunication (JIST)***, Vol. 6, No. 4, (2018), 204-208. https://doi.org/10.7508/jist.2018.04.003
74. Dempster, A.P., Laird, N.M. and Rubin, D.B., "Maximum likelihood from incomplete data via the em algorithm," ***Journal of the Royal Statistical Society: Series B (Methodological)***, Vol. 39, No. 1, (1977), 1-22. https://doi.org/10.1111/j.2517-6161.1977.tb01600.x

Persian Abstract

چکیده

الگوریتم‌های بهینه‌سازی فراابتکاری گونه‌های نسبتا جدیدی از الگوریتم‌های بهینه‌سازی هستند که برای مسائل بهینه‌سازی دشوار که نمی‌توان از روش‌های کلاسیک برای آنها استفاده کرد کاربرد فراوانی دارند و روشی شناخته‌شده و بسیار گسترده برای مسائل دشوار بهینه‌سازی به حساب می‌آیند. در این مقاله یک الگوریتم بهینه‌سازی فراابتکاری جدید که ایده‌ی اصلی آن از یک نوع حرکت در فیزیک ایده‌برداری شده‌است ارائه می‌شود تا به وسیله‌ی آن بتوان به نتایج بهتری نسبت به سایر الگوریتم‌های بهینه‌سازی در این حوزه دست یافت و برای رسیدن به نقطه‌ی مطلوب‌تر مسیر جدیدی را تجربه کرد. از این رو در این مقاله بعد از معرفی الگوریتم بهینه‌سازی پرتابه‌ها، ابتدا این الگوریتم بهینه‌سازی را با الگوریتم‌های شناخته شده و قدرتمند این حوزه و بر روی توابع سنجش معروف، مقایسه کردیم و در ادامه، عملکرد الگوریتم بهینه‌سازی پرتابه‌ها را در دو مورد از کاربردهای عملی با الگوریتم‌های دیگر مورد سنجش قرار دادیم و نتایج این مقایسه‌ها را در جدول‌ها و شکل‌های گوناگون و برای حالت‌های مختلف آورده‌ایم که نشان‌دهنده‌ی دقت به مراتب بیشتر الگوریتم بهینه‌سازی پرتابه‌ها نسبت به الگوریتم‌های دیگر می‌باشد.