Inteligencia Artificial 28 (75) (2025), 114-139
doi: 10.4114/intartif.vol28iss75pp114-139

INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

Two competitive hybridization approaches based on combining of Giza Pyramids Construction with Particle Swarm Optimization for solving global optimization problems

Sasan Harifi[1,A], Faraz Davachi[1,B], Narges Mohammadi[1,C], Soheil FaridMohammadzadegan[1,D]

[1] Department of Computer Engineering, Karaj Branch, Islamic Azad University, Karaj, Iran

[A] s.harifi@kiau.ac.ir (Corresponding author)
[B] farazdavachi23@gmail.com
[C] nargesmohammadii@protonmail.com
[D] soheil.faridmohammadzadegan@kiau.ac.ir

Abstract. Optimization problems are complex problems that are very difficult to solve. Although these types of problems are solved in the real world using exact methods, these methods are very time-consuming and costly. By using soft computing methods, the time and cost of problem-solving can be reduced to some extent. Engineering problems are among the complex real-world problems that can be solved through soft computing methods. One of these methods is the use of metaheuristic algorithms to optimize the solution of these types of problems. The Particle Swarm Optimization (PSO) algorithm is a common and state-of-the-art metaheuristic algorithm used to solve engineering optimization problems. This algorithm is known as swarm-based optimization techniques and has a very powerful mathematical basis. Another recently published algorithm is the Giza Pyramids Construction (GPC) algorithm. The GPC algorithm models the technological advancements of construction in ancient times. Both algorithms have many advantages through which optimization problems can be solved effectively. To increase the power of metaheuristic algorithms and solve optimization problems more effectively through them, the idea of competitive hybridization algorithms has been proposed. In this paper, two competitive hybrid approaches of combining PSO and GPC algorithms are presented. These two competitive hybridization approaches have been first applied to 45 benchmark functions and have been evaluated and analyzed statistically. Then they have been applied to six classic engineering problems. Algorithms presented in each step have been compared with Genetic Algorithm (GA), PSO, some recently published algorithms, and their combined approaches. The results of experiments and statistical analysis show that the solution to engineering problems has been done more effectively by using the two proposed combinations.

Keywords: Optimization, Metaheuristic, Giza Pyramids Construction algorithm, Particle Swarm Optimization, Hybrid metaheuristic algorithm, Engineering problems.

1 Introduction

The optimization problem refers to problems that are very difficult to solve in the real world due to their high complexity [1]. Almost every problem that exists around us can be an optimization problem, many of which have complexities and problems that require special computational approaches to solve them. Optimization problems are more common in engineering, economics, and mathematics than in other fields. The main parts of the optimization problem are decision variables, constraints in the problem, and objective functions [2]. In these problems, we mainly seek to provide solutions to reduce or eliminate the limitations of the problem, including technical limitations, time, laws, etc. Over time, optimization problems are also becoming more complex.

ISSN: 1137-3601 (print), 1988-3064 (on-line)

As mentioned, there are many optimization problems in engineering fields. For example, in electrical engineering, solving and reducing limitations in increasing power losses, checking voltage instability and line overload can be optimization problems. Also, the optimization of systems and voltage in power transmission and power flow control is one of the optimization problems in electrical engineering [3]. In civil engineering, one of the most important issues and challenges is the design and optimization of structures, in which optimization methods are used to increase efficiency based on mathematical programming techniques, reduce energy consumption and pollution, reduce impurities, and so on to find the best solution and result [4]. The training of machine learning models is also considered a type of optimization problem, so optimization plays a significant role in machine learning models [5]. Their goal is to optimize the training error and measure fitness and mutual entropy, which by finding the best element in the objective function and optimizing the model, enable the training and learning of the entire model efficiently [6].

Optimization problems can be solved in two deterministic and stochastic ways. The deterministic method uses the derivatives of the objective functions, which are used for linear, convex, and simple problems. The random method and optimization algorithms use random scanning of the search space and random operators instead of using derivatives of functions. In optimization algorithms to solve optimization problems, among the generated and improved candidate solvable solutions, after performing the steps of the algorithm, the best ones are selected as the solution. These algorithms require less time and calculations compared to deterministic algorithms. A group of optimization algorithms that have been of great interest in the last decade are metaheuristic optimization algorithms. Due to the high adaptability of metaheuristic algorithms in solving optimization problems, they have become one of the most popular methods for solving these problems [7].

Metaheuristic algorithms have different categories. An important category that is also of interest to researchers is the nature-inspired category. The algorithms presented in this category model simple behaviors and laws of nature. One of the behaviors found in nature is crowding behavior. In this behavior, the population tries to reach a common goal by helping each other [8]. Another category that has been introduced recently is the category of algorithms inspired by ancient. Algorithms presented in this category mainly model the technological advances of ancient times that were beyond their time [9]. Of course, each algorithm has its strengths and weaknesses. For this reason, the idea of combining algorithms or hybridization is proposed. Algorithms can be combined to take advantage of their strengths. As a result, the weak points are compensated and a better overall performance is obtained. Each algorithm can have its unique approach or focus, and combining them leads to advanced analysis or problem-solving. The combination of algorithms provides the possibility of using different perspectives, dealing with complexity, improving accuracy, and increasing robustness [10].

Some advantages of combining algorithms can be mentioned. The combination of algorithms makes it possible to detect and correct errors or outliers more effectively. If an algorithm produces incorrect results or is sensitive to certain data types, combining it with other algorithms can help identify and resolve these problems and improve overall robustness. Some problems may be too complex for an algorithm to solve efficiently. Combining algorithms that specialize in different aspects or phases of the problem can lead to more comprehensive solutions and better resolution of complexity. Combining algorithms provides great flexibility to changing conditions or new data. If an algorithm is inefficient or outdated, by merging it with a new algorithm, it is possible to ensure that an up-to-date and efficient approach is created [11].

The Giza Pyramids Construction (GPC) algorithm is a powerful population-based algorithm inspired by ancient times. This algorithm simulates the optimal construction method of pyramids. Previously, this algorithm alone has been used in solving various problems such as energy management [12], knapsack [13], evacuation models [14], network reliability [15], color thresholding [16], increasing system security [17], reactive power distribution [18], and so on. In this paper, we present two improved versions of GPC by combining it with the Particle Swarm Optimization (PSO) algorithm, which is an algorithm inspired by nature. In this way, the concepts of individual understanding and social learning in the PSO algorithm are added to the basic concepts of the GPC algorithm.

The main goal of this paper is to achieve the optimal values of the criterion functions and to improve the standard GPC algorithm by combining its concepts with the concepts in the PSO algorithm based on competitive hybridization approach. Therefore, two competitive hybrid PSOGPC and GPCPSO algorithms are used to validate and compare the quality of the solution. In addition, the main contribution of the paper is the use of two proposed competitive hybrid PSOGPC and GPCPSO algorithms to reduce computational costs, reach the most optimal possible solution, and compare it with other improved methods and common hybrid algorithms in solving engineering problems. In summary, the innovation of the paper is as follows:

- Two new hybrid approaches has been proposed and developed.
- Usage fast convergence ability of PSO and the efficient local search ability of GPC.
- The convergence patterns of algorithms have been exhaustively explored and validated.
- Six engineering problems were raised and proposed algorithms were applied on them.

- Competing algorithms were considered of hybrid metaheuristics.

The rest of the paper is structured as follows: Section 2 presents the related works. Section 3 describes the combinations of GPC with PSO in detail. Section 4 is experimental results. Section 5 provides statistical analysis. Section 6 demonstrates solving classical engineering problems. Finally, section 7 expresses the conclusions.

2 Related works

In the literature, there are many studies in the field of hybridizing metaheuristic algorithms. Mohammadzadeh et al [19] presented the combination of Ant Lion Optimizer (ALO) and Sine Cosine Algorithm (SCA). They used their algorithm to solve the scientific workflow scheduling problem. The main goal of this algorithm is to improve search performance by using cluster algorithms and using random numbers based on chaos theory in a green cloud computing environment. This algorithm tries to reduce the minimum run time and cost of performing tasks, reduce energy consumption and create a green environment for cloud computing. Also, increasing the efficiency rate was another goal that the authors were looking for. Jain and Sharma [20] presented a combination of the Slap Swarm Algorithm (SWA) and Grey Wolf Optimizer (GWO) for parameter tuning in cloud computing. The results of their experiments show that these algorithms perform better than other algorithms that were compared and improve resource planning and parameter setting in the cloud environment. On the other hand, cloud data storage meets users' needs with infinite scalability, more reliable storage, and lower-cost models. One of the challenging issues in this field is the large amount of data that affects the quality. Tahir et al [21] proposed an efficient storage model with the approach of combining the Pelican Optimization Algorithm (POA) with the Billiards Optimization Algorithm (BOA) for the mentioned challenge. In their proposed system for storage, cloud data has been stored and used for system performance analysis, and due to the extensiveness of the data, different data sets and different storage devices were used. To reduce the limitations in the data allocation rules and the capacity of the devices, their presented hybrid approach has been used.

One of the subjects in which the combination of metaheuristic algorithms has been used a lot is feature selection. Dey et al [22] used CNN and a metaheuristic hybrid feature selection algorithm in which the Manta Ray Foraging Optimization (MRFO) algorithm is combined with the Golden Ratio Optimization Method (GROM) algorithm to diagnose the disease of COVID-19. To improve disease diagnosis, they used chest computed tomography images. They claim that their proposed hybrid method provided high accuracy. Kareem et al [23] presented a new feature selection method that works by enhancing the performance of the Gorilla Troops Optimizer (GTO) based on the Bird Swarms Algorithm (BSA). They applied their hybrid algorithm in optimizing feature selection. Al-Maqbali [24] has combined the Wolf Pack Algorithm (WPA) into Particle Swarm Optimization (PSO) in his paper. He applied his method of optimal selection of features and adjustment of neural network weights. With this, he tried to improve the accuracy of breast cancer diagnosis. Das et al [25] combined Bat Algorithm (BA) with Hill-Climbing in their research. They also used their hybrid algorithm to improve feature selection. Bhattacharyya et al [26] used the combination of the Mayfly Algorithm (MA) and Harmony Search (HS) to optimize feature selection. Yan et al [27] proposed the hybridization of The Coral Reefs Optimization (CRO) with Simulated Annealing (SA). Mafarja and Mirjalili [28] combined the Whale Optimization Algorithm (WOA) and SA. They also used their combined algorithm to improve feature selection. The WOA is inspired by the well-known behaviour of whales and is used as a swarm intelligence optimization algorithm. This algorithm has attracted the attention of many researchers by providing new ideas to improve efficiency and application. However, WOA has problems such as slow convergence, low accuracy in optimization, and the tendency to collapse in local optimal, which can significantly affect the performance of the algorithm. To improve the performance of this algorithm, Tang et al [29] combined it with the Artificial Bee Colony (ABC) algorithm. Their experiments show that the hybrid algorithm presented by them has a good ability in solving complex problems.

In the field of optimization, some algorithms such as GWO and PSO are very important. These two algorithms each have a unique search mechanism. Zhang et al [30] have combined these two algorithms. Although PSO has good efficiency in optimization, it often reaches a local minimum. On the other hand, despite the high efficiency of GWO in optimizing problems, there are some problems such as the lack of sufficient search power on a global scale. The combination of these two algorithms was done to solve the mentioned problems of these two algorithms. Sohouli et al [31] used the combination of the Genetic Algorithm (GA) and PSO to estimate Magnetic Anomaly Parameters. In their method, the PSO algorithm enhances the magnetic data. This is while the GA changes the decision to evaluate the model parameters. In addition, the balance between exploration and exploitation abilities by integrating genetic operators in the hybrid PSO-GA algorithm has enhanced the performance of this method. The results of their experiments show that their proposed method can provide valuable results in estimating model parameters. They believe that the method presented by them can be used as an effective tool for exploratory

geological studies and estimation of natural resources. Goodarzian et al [32] developed a mixed integer linear programming (MILP) formulation for the production-distribution-routing problem in a sustainable agricultural supply chain network, in which the combination of SA and PSO algorithms, as well as the combination of GA and Tabu Search (TS), were used. Khan et al [33] combined GA with PSO to solve the model of process planning in a reconfigurable manufacturing system. They found that hybrid metaheuristic algorithms help to balance the search of the problem-solving space. Gupta and Deep [34] combined GWO with Deferential Evolution (DE). They claim that the proposed algorithm has significantly enriched the standard GWO algorithm. Jahannoosh et al [35] combined the GWO algorithm with the Sine Cosine Algorithm (SCA). They used their combined algorithm for the optimal design of renewable energy systems such as solar cells, wind turbines, and fuel cells. They claim that the presented method by them, has more speed and accuracy of convergence than other methods. The goal of optimal design in these systems is more reliability. Kamboj [36] proposed a hybrid algorithm of PSO and GWO to solve the unit commitment problem. Also, Teng et al [37] proposed a GWO algorithm in combination with PSO. In their algorithm, they used the idea of PSO global and personal best to update the position of each wolf. Their simulation results show that the proposed algorithm can have an acceptable global optimal solution and better robustness than other algorithms.

Due to the development and popularity of the Internet, the majority of people tend to buy and sell online, and many platforms and applications have been built and developed for this purpose. Most of the purchases are made on the internet and it goes from offline to online shopping. The tire industry is one of the applications that today has become inclined towards online shopping. Fathollahi-Fard et al [38] have modeled the tire industry as an optimization problem. They then proposed a fuzzy approach and two hybrid metaheuristic algorithms to deal with uncertain problem parameters such as prices and demand. According to their suggestion, Red Deer Algorithm (RDA) and WOA are hybridized with the GA and SA. Sengathir et al [39] combined ABC and Firefly Algorithm (FA) to improve and optimize the distance and lower delay between nodes of wireless sensor networks. The results of their experiments show that their approach can increase the stability time of homogeneous and heterogeneous networks. Al-Otaibi et al [40] conducted a study on energy efficiency in wireless sensor networks, in which a hybrid algorithm combining Water Wave Optimization (WWO) with a hill-climbing algorithm was used. They used their algorithm to improve network routing performance. Their experiments showed improved network performance. Agnihotri et al [41] used hybrid PSO with GA for routing in the wireless sensor network. Harifi et al [42] presented a combination of the Emperor Penguins Colony (EPC) algorithm and GA. They added two GA operators namely crossover and mutation to the EPC algorithm. The combined algorithm provided by them was used for community detection in social networks. They showed that genetic algorithm operators greatly improve the performance of the EPC algorithm.

In other studies, Kumar et al [43] proposed the combination of the Grasshopper Optimization Algorithm (GOA) and Black Hole (BH) algorithms in their research. They applied their approach to privacy in the Industrial Internet of Things (IIoT). Also in another study, Akhtar et al [44] combined two algorithms Spotted Hyena Optimization (SHO) and Sun Flower Optimization (SFO) for optimization in the discussion of the Internet of Things (IoT). In other researches, Seydanlou et al [45] combined Virus Colony Search (VCS) algorithm and SA. The aim was to use the combination of these algorithms for the optimal and sustainable design of supply chain networks for product packaging. Hu et al [46] proposed a hybrid algorithm that is the Squirrel Search Algorithm (SSA) in combination with Invasive Weed Optimization (IWO). The Krill Herd (KH) algorithm combined with HS by Abualigah et al [47]. In a review study, Bouaouda and Sayouti [48] investigated combined metaheuristic algorithms for improving the sustainability of energy production infrastructures in hybrid renewable energy systems. According to their research, using the combination of the SA algorithm and population-based metaheuristic algorithms is more reliable than other methods for solving the problem due to less computation. Khalili-Goudarzi et al [49] has combined SA and Gravitational Search Algorithm (GSA) for multi-product oil pipeline scheduling problem. Verma and Parouha [50] used an advanced hybrid algorithm that combines DE algorithm and PSO algorithm for engineering design optimization problems such as structural engineering. Their new combination algorithm has a multi-population approach and ensures the diversity of the population and also establishes a proper balance between exploration and exploitation due to having control parameters. Kumar and Vidyarthi [51] combined PSO and GA algorithms for directed acyclic graph programming with communication in multiprocessor systems. They showed that their hybrid algorithm was very effective in the directed acyclic graph problem.

The Butterfly Optimization Algorithm (BOA) is simple and efficient. This algorithm depends on the mechanism of searching for food in butterflies, which results from some kind of interaction between butterflies and their food. One of the problems of BOA is its tendency towards local optimal. This means that the algorithm may tend to search for local optimal points in its initial iterations and avoid searching for global optimal points. To solve this problem, Wang et al [52] hybridized the BOA algorithm with the Flower Pollination Algorithm (FPA). The FPA algorithm has good exploration ability and its hybridization with the BOA algorithm has improved the algorithm's exploration

ability. They applied their hybrid algorithm to solve classical engineering problems and obtained good results. Aydilek [53] has conducted a study concerning the reduction of computationally expensive numerical problems, in which he suggests the combination of the PSO algorithm and FA. The proposed hybrid algorithm can use the strengths of both PSO and FA well. Goel and Maini [54] developed a hybrid algorithm that incorporates certain aspects of FA and Ant Colony Optimization (ACO) algorithms to solve a class of vehicle routing problems. The performance of their proposed algorithm compared to some other approaches showed that the hybrid algorithm proposed by them was successful in solving the problem. Wahid and Ghazali [55] combined FA with Pattern Search (PS) algorithm. Rashid and Abdullah [56] have used a combination of an ABC algorithm and the GA in their paper. They then applied their combined algorithm to the Back Propagation Neural Network (BPNN) for the classification application, and diabetes diagnosis. Sheng et al [57] used the combination of the ACO algorithm with GA to improve takeout delivery schedule. They then tested their approach on a real model and reported satisfactory results. Mzili et al [58] combined GA with Spotted Hyena Optimization (SHO) algorithm to solve the production shop scheduling problem. They compared their algorithm with other state-of-the-art optimization algorithms. They showed that their algorithm was very effective in solving this problem. Mzili et al [59] also combined GA with penguin search optimization (PSeOA) for solving flow shop scheduling problem. This algorithm combines the genetic diversity of GA with the fast convergence of PSeOA. Its method shows significant improvements in solving the flow shop scheduling problem.

3 Combinations of GPC with PSO

In this section, both the base and standard algorithms namely GPC and PSO are first described in separate subsections so that the inspiration, formulation, and method of using the algorithm are described. Then, in subsection 3.3, the method of combining these two algorithms is described, so that once the PSO algorithm is considered as the base algorithm and again the GPC Algorithm is considered as the base algorithm. In this way, two completely independent combined algorithms are presented.

3.1 Standard GPC

The meaning of ancient times is the time when writing was used to record historical events. To understand this era, two sources are available: Archaeologists and textual sources. These sources show that although there were many limitations in ancient times, these limitations did not prevent the development of technology and engineering. We can find out by studying civilizations and engineering methods related to construction and related technologies that they have always been looking for ways to use a better way and a better way of life. As evidence of this, the existence of heritage and architecture left over from this era can be mentioned. For the first time, the Giza Pyramids Construction (GPC) [60] algorithm took this era as its source of inspiration. This algorithm is inspired by the construction method of pyramids. The way of moving the stone pieces by the workers on the ramp to build the pyramids is modeled in this algorithm. What has been important and influential in building the pyramids are factors such as ramps, management of workers, friction, and slope. Based on this algorithm, a worker who works better and more effectively than others can reach the special agent title of pharaoh and therefore other workers will be compared with him. There has always been competition among workers to reach this special position. In addition, the workers whose work is not good will be substituted by other workers. Based on this algorithm, even the weakest worker can achieve the special position of pharaoh by strengthening himself. Figure 1 shows the pseudo-code of the standard GPC algorithm.

Algorithm 1: Pseudo-code of the standard Giza Pyramids Construction (GPC) Algorithm.

```
STEP 1:
   generate initial population of stone blocks or workers;
   generate position and cost of stone block or worker;
   determine best worker as Pharaoh's agent;
   STEP 2: for FirstIteration to MaxIteration do
      STEP 3: for i=1 to n do (all n stone blocks or workers)
         calculate amount of stone block displacement (Eq. 1);
         calculate amount of worker movement (Eq. 2);
         estimate new position (Eq. 3);
         investigate possibility of substituting workers (Eq. 4);
         determine new position and new cost;
         if new_cost < Pharaoh's agent cost then
               set new_cost as Pharaoh's agent cost;
         end if
      END STEP 3
      Sort solutions for next iteration;
   END STEP 2
END STEP 1
```

Fig 1. Pseudo-code of the standard GPC algorithm

In this algorithm, it is assumed that the stone block is pushed upwards on the ramp with an initial speed of v_0. After traveling the path d by the stone block, the block stops. Figure 2 shows the force acting on the stone.

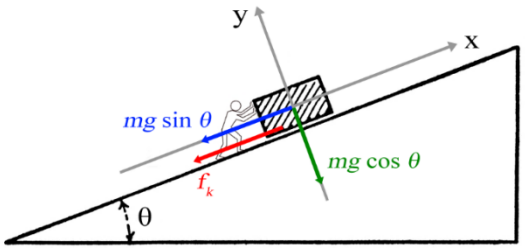

Fig 2. Forces acting on the stone block

Considering the conditions in the algorithm, it is necessary to have a time-independent equation in movement with constant acceleration. Therefore, the distance traveled by the stone on the ramp can be obtained by the following equation,

$$d = \frac{v_0^2}{2g(\sin\theta + \mu_k \cos\theta)} \tag{1}$$

where in this equation, d is the path traveled by the stone with worker. g is equal to 9.8 and Earth's gravity. The ramp makes an angle with the horizon, denoted by θ. The initial speed of the stone is indicated by v_0 and in each iteration it can be produced with the uniformly distributed random number, that is $v_0 = rand(0,1)$. In addition, the coefficient of kinetic friction between the stone and the ramp (μ_k) is determined by $\mu_k = rand(\mu_{k\_min}, \mu_{k\_max})$, which means a uniform random distribution.

If the worker who pushes the stone wants to have a better mastery of this work, it must perform movements that the following equation shows the amount of movement of the worker when pushing the stone.

$$x = \frac{v_0^2}{2g\sin\theta} \tag{2}$$

where the above equation is the Equation 1 without friction. If these two equations are combined then the new equation below is obtained which shows the same changes of stone and worker.

$$\vec{p} = (\vec{p_i} + d) \times x\overrightarrow{\epsilon_i} \tag{3}$$

where p_i represents the current position and d is the value of the path of moving the stone, x is the amount of movement of the worker, ϵ_i is a vector of random numbers that can be based on Uniform, Normal, or Lévy distribution.

In the standard GPC algorithm, there is a fifty percent probability that a worker will be substituted by another worker, and this will lead to a balance in exploration and exploitation performance. However, the percentage of this possibility has been changed in different works done by researchers. This operation can be done by the following equation,

$$\zeta_k = \begin{cases} \psi_k, \text{if } rand[0,1] \leq 0.5 \\ \varphi_k, \text{otherwise} \end{cases} \quad (4)$$

where in the above relationship, ψ is the solution generated in the previous equation (Equation 3), φ is the original solution, and ζ will show the new solution.

3.2 Standard PSO

Particle Swarm Optimization (PSO) is one of the most widely used optimization algorithms in the world of engineering and computer science. The main idea of this algorithm is inspired by the group behavior of birds, which exchange information and work together to find the best solutions. It is assumed that a group of birds randomly searches for food in search space. There is only one piece of food in the search space. None of the birds know where the food is. One of the best strategies can be to follow the bird that has the shortest distance to the food. Each particle or bird has a fitness value that is calculated by a fitness function. The closer the particle is to the target or food in the search space in the bird movement model, the more merit it has. The pseudo-code of the PSO algorithm is shown in Figure 3.

Algorithm 2: Pseudo-code of the standard Particle Swarm Optimization (PSO) Algorithm.

```
STEP 1:
   generate initial population (swarm size);
   generate position and cost of population;
   determine personal best (pBest) and global best (gBest) of population;
   STEP 2: for FirstIteration to MaxIteration do
      STEP 3: for i=1 to n do (all n particles)
         calculate velocity (Eq. 5);
         estimate new position (Eq. 6);
         determine new cost based on new position;
         if new cost < pBest then
               set new cost as pBest cost;
               if pBest cost < gBest cost then
                     set pBest cost as gBest cost;
               end if
         end if
      END STEP 3
      store solutions for next iteration;
   END STEP 2
END STEP 1
```

Fig 3. Pseudo-code of the standard PSO algorithm

In this algorithm, each particle has a position and a speed and moves in the search space. Particles improve their search mechanism by observing each other's position and performance. At each iteration of the algorithm, the particles receive the position information of the global best particle and update their personal best position. This information exchange to the most optimal point has a positive effect on the evolution of the algorithm. Particles are updated using two formulas, one for velocity and one for the position, so that for the velocity we have,

$$v(t+1) = wv(t) + r_1 c_1 \big(pBest(t) - x(t)\big) + r_2 c_2 \big(gBest(t) - x(t)\big) \quad (5)$$

where w is the coefficient of velocity, which is also called inertia, and x is the current position. Also, $r_1 c_1$ is a coefficient of the distance of the personal best from the current point, which is generated through the multiplication of a random number r_1 and a number related to learning c_1. In addition, $r_2 c_2$ is a coefficient of the distance of the global best from the current point, where r_2 is a random number that is multiplied by c_2, which is a learning coefficient. For the position of the particle, there is,

$$x(t+1) = x(t) + v(t+1) \quad (6)$$

where x is the current position and v is the velocity obtained from the previous equation.

3.3 Proposed competitive hybridization

In this subsection, we describe the two proposed competitive hybridization algorithms. In general, the algorithms that are created by combining two or more algorithms can be divided into three categories. These three categories are sequential combination algorithms, parallel combination algorithms, and merged combination algorithms [42]. In the category of sequential combination algorithms, one of the algorithms is implemented first, and then the second algorithm is applied to the results obtained from the first algorithm. In the category of parallel combination algorithms, two or more algorithms are applied simultaneously to the same population. In the category of merged combination algorithms, different parts of algorithms are combined with each other. It can be the use of different operators of one algorithm in another algorithm.

In this paper, the first category namely sequential combination is used. For this purpose, two competitive PSOGPC and GPCPSO algorithms are presented. In the PSOGPC algorithm, in each iteration, the PSO algorithm is first executed and produces a new solution. Then the GPC algorithm is applied to the solutions of the PSO algorithm. The GPCPSO algorithm, is exactly the opposite of the PSOGPC algorithm, in this way that the GPC algorithm is executed first, then the PSO algorithm is applied to the solutions obtained from the GPC. The main goal is that these two hybrid algorithms are tested simultaneously in order to obtain the best solution based on the competitiveness approach for the desired problems. However, it should be noted that these two algorithms are completely separate and independent of each other, each with its combined approach to optimization and problem-solving. However, it is possible to run these two algorithms in such a way that the outputs obtained are compared simultaneously and the best output is selected. The output of each algorithm is specific to that algorithm and is not used as a comparative solution for the next iteration of the other algorithm. Figure 4 shows the pseudo-code of combined PSOGPC and GPCPSO algorithms. Also, Figure 5 shows the representation scheme of PSOGPC and GPCPSO algorithms.

Algorithm 3: Pseudo-code of the proposed PSOGPC.

```
STEP 1:
   generate initial population (swarm size);
   generate position and cost of population;
   determine personal best (pBest) and global best (gBest);
   STEP 2: for FirstIteration to MaxIteration do
      STEP 3: for i=1 to n do (all n particles)
         calculate velocity (Eq. 5);
         estimate new position (Eq. 6);
         determine new cost based on new position;
         if new cost < pBest then
               set new cost as pBest cost;
               if pBest cost < gBest cost then
                     set pBest cost as gBest cost;
               end if
         end if
      END STEP 3
      store solutions for next algorithm (GPC algorithm);
      STEP 4: for i=1 to n do (all n stone blocks or workers)
         calculate amount of stone block displacement (Eq. 1);
         calculate amount of worker movement (Eq. 2);
         estimate new position (Eq. 3);
         investigate possibility of substituting workers (Eq. 4);
         determine new position and new cost;
         if new_cost < Pharaoh's agent cost then
               set new_cost as Pharaoh's agent cost;
         end if
      END STEP 4
      sort solutions for next iteration;
   END STEP 2
END STEP 1
```

Algorithm 4: Pseudo-code of the GPCPSO.

```
STEP 1:
   generate initial population of stone blocks or workers;
   generate position and cost of stone block or worker;
   determine best worker as Pharaoh's agent;
   STEP 2: for FirstIteration to MaxIteration do
      STEP 3: for i=1 to n do (all n stone blocks or workers)
         calculate amount of stone block displacement (Eq. 1);
         calculate amount of worker movement (Eq. 2);
         estimate new position (Eq. 3);
         investigate possibility of substituting workers (Eq. 4);
         determine new position and new cost;
         if new_cost < Pharaoh's agent cost then
               set new_cost as Pharaoh's agent cost;
         end if
      END STEP 3
      sort solutions for next algorithm (PSO algorithm);
      STEP 4: for i=1 to n do (all n particles)
         calculate velocity (Eq. 5);
         estimate new position (Eq. 6);
         determine new cost based on new position;
         if new cost < pBest then
               set new cost as pBest cost;
               if pBest cost < gBest cost then
                     set pBest cost as gBest cost;
               end if
         end if
      END STEP 4
      store solutions for next iteration;
   END STEP 2
END STEP 1
```

Fig 4. The pseudo-code of competitive hybrid PSOGPC and GPCPSO algorithms. Left is PSOGPC and right is GPCPSO.

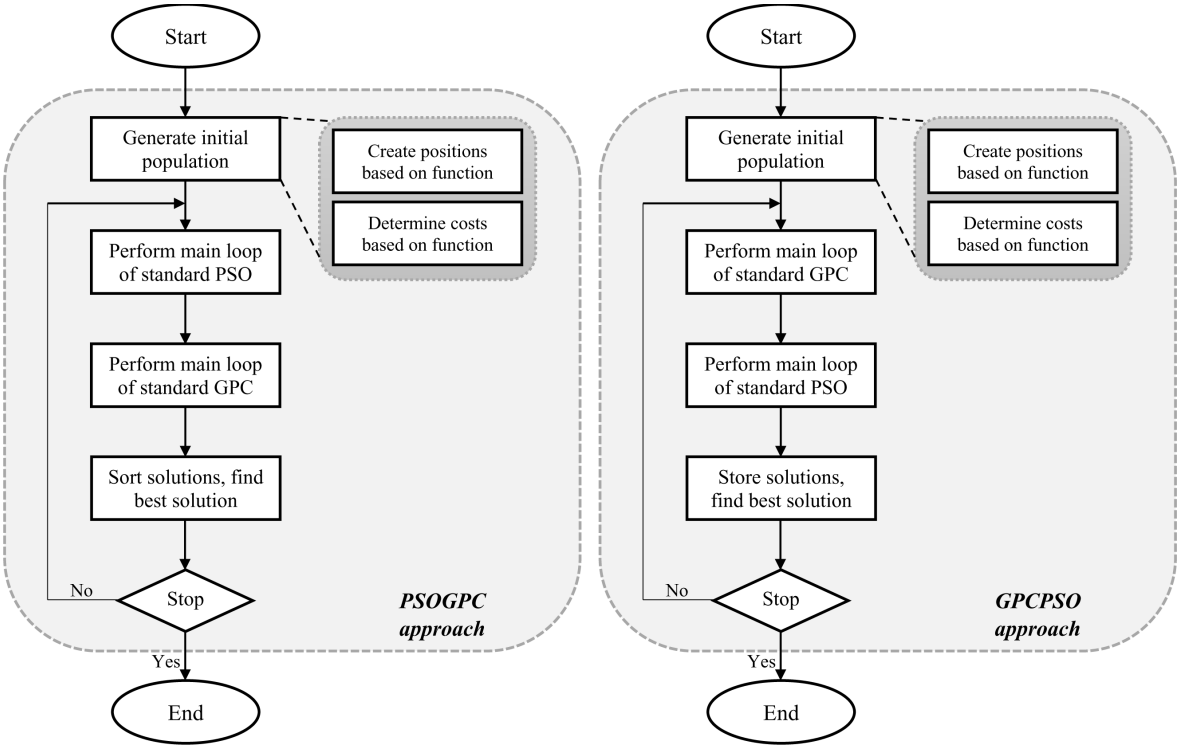

Fig 5. The representation scheme of competitive hybrid PSOGPC and GPCPSO algorithms

4 Experimental results

In this section, the details of the experiments are described. Forty-five standard benchmark test functions have been used to evaluate the performance of the proposed hybrid algorithms. Also, the proposed hybrid algorithms have been compared with the standard GA, standard PSO, and standard GPC, as well as two hybrid algorithms, namely GAPSO and PSOGA. Benchmark functions include two categories, unimodal and multimodal, which are introduced in Table 1 and Table 2, respectively. The selected functions are quite diverse so that they cover from one dimension to d-dimension. They also include simple functions and complex and very complex functions, which are very useful for evaluating the properties of an optimization algorithm.

Table 1. Unimodal benchmark test functions considered to compare algorithms

Function	Equation	Range	Dim	$f(x^*)$
Bohachevsky	$f_1(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$	$[-100, 100]$	2	0
Booth	$f_2(x) = (x_1 + 2x_2 + 7)^2 + (2x_1 + x_2 + 5)^2$	$[-10, 10]$	2	0
Easom	$f_3(x) = -\cos(x_1)\cos x_2 \exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$	$[-100, 100]$	2	-1
Gramacy & Lee	$f_4(x) = \frac{\sin(10\pi x)}{2x} + (x-1)^4$	$[0.5, 2.5]$	1	-0.8690
Griewank	$f_5(x) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]$	d	0
Hyper-Ellipsoid	$f_6(x) = \sum_{i=1}^{d} \sum_{j=1}^{i} x_j^2$	$[-65.536, 65.536]$	d	0
Matyas	$f_7(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$	$[-10, 10]$	2	0
Perm	$f_8(x) = \sum_{i=1}^{d} \left(\sum_{j=1}^{d} (j+\beta)\left(x_j^i - \frac{1}{j^i}\right)\right)^2$	$[d, d]$	d	0
Power Sum	$f_9(x) = \sum_{i=1}^{d} \left[\left(\sum_{j=1}^{d} x_j^i\right) - b_i\right]^2$ where $b = (8,18,44,114)$	$[0, 4]$	4	0
Schaffer N.2	$f_{10}(x) = 0.5 + \frac{\sin^2(x_1^2 - x_2^2) - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2}$	$[-100, 100]$	2	0
Schaffer N.4	$f_{11}(x) = 0.5 + \frac{\cos(\sin(\lvert x_1^2 - x_2^2 \rvert)) - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2}$	$[-100, 100]$	2	0.2925
Sphere	$f_{12}(x) = \sum_{i=1}^{d} x_i^2$	$[-5.12, 5.12]$	d	0
Sum-Powers	$f_{13}(x) = \sum_{i=1}^{d} \lvert x_i \rvert^{i+1}$	$[-1, 1]$	d	0
Sum-Squares	$f_{14}(x) = \sum_{i=1}^{d} i x_i^2$	$[-10, 10]$	d	0

Zakharov	$f_{15}(x) = \sum_{i=1}^{d} x_i^2 + \left(\sum_{i=1}^{d} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{d} 0.5ix_i\right)^4$	$[-5, 10]$	d	0

Table 2. Multimodal benchmark test functions considered to compare algorithms

Function	Equation	Range	Dim	$f(x^*)$
Ackley	$f_{16}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + \exp$	$[-32.768, 32.768]$	d	0
Beale	$f_{17}(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$	$[-4.5, 4.5]$	2	0
Branin	$f_{18}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	$[-5,10]$	2	0.3978
Bukin	$f_{19}(x) = 100\sqrt{\lvert x_2 - 0.01x_1^2\rvert} + 0.01\lvert x_1 + 10\rvert$	$[-3,3]$	2	0
Camel Six-Hump	$f_{20}(x) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$	$[-3,3]$	2	-1.0316
Camel Three-Hump	$f_{21}(x) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1x_2 + x_2^2$	$[-5,5]$	2	0
Colville	$f_{22}(x) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2 + 10.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1)$	$[-10,10]$	4	0
Cross-In-Tray	$f_{23}(x) = -0.0001\left(\left\lvert\sin(x_1)\sin(x_2)\exp\left(\left\lvert 100 - \frac{\sqrt{x_1^2+x_2^2}}{\pi}\right\rvert\right)\right\rvert + 1\right)^{0.1}$	$[-10, 10]$	2	-2.0626
De Jong	$f_{24}(x) = \left(0.002 + \sum_{i=1}^{25}\frac{1}{i+(x_1-a_{1i})^6+(x_2-a_{2i})^6}\right)^{-1}$	$[-65.536,65.536]$	2	0
Dixon Price	$f_{25}(x) = (x_1 - 1)^2 + \sum_{i=2}^{d} i(2x_i^2 - x_{i-1})^2$	$[-10, 10]$	d	0
Drop-Wave	$f_{26}(x) = -\frac{1+\cos\left(12\sqrt{x_1^2+x_2^2}\right)}{0.5(x_1^2+x_2^2)+2}$	$[-5.12, 5.12]$	2	-1
Eggholder	$f_{27}(x) = -(x_2 + 47)\sin\left(\sqrt{\left\lvert x_2 + \frac{x_1}{2} + 47\right\rvert}\right) - x_1\sin\left(\sqrt{\lvert x_1 - (x_2 + 47)\rvert}\right)$	$[-512, 512]$	2	-959.6407
Forrester	$f_{28}(x) = (6x - 2)^2\sin(12x - 4)$	$[0,1]$	1	-6.0207
Hartmann 3D	$f_{29}(x) = -\sum_{i=1}^{4}\alpha_i\exp\left(-\sum_{j=1}^{3}a_{ij}(x_j - p_{ij})^2\right)$ where α, a, p are from [61]	$[0,1]$	3	-3.8628
Hartmann 4D	$f_{30}(x) = \frac{1}{0.839}\left[1.1 - \sum_{i=1}^{4}\alpha_i\exp\left(-\sum_{j=1}^{3}a_{ij}(x_j - p_{ij})^2\right)\right]$ where α, a, p are from [61]	$[0,1]$	4	-3.1345
Hartmann 6D	$f_{31}(x) = -\sum_{i=1}^{4}\alpha_i\exp\left(-\sum_{j=1}^{6}a_{ij}(x_j - p_{ij})^2\right)$ where α, a, p are from [61]	$[0,1]$	6	-3.3223
Holder-Table	$f_{32}(x) = -\left\lvert\sin(x_1)\cos(x_2)\exp\left(\left\lvert 1 - \frac{\sqrt{x_1^2+x_2^2}}{\pi}\right\rvert\right)\right\rvert$	$[-10,10]$	2	-19.2085
Langermann	$f_{33}(x) = \sum_{i=1}^{5}c_i\exp\left(-\frac{1}{\pi}\sum_{j=1}^{d}(x_j - a_{ij})^2\right)\cos\left(\pi\sum_{j=1}^{d}(x_j - a_{ij})^2\right)$ where c, a are from [61]	$[0,10]$	2	-1.4
Levy	$f_{34}(x) = \sin^2(\pi\omega_1) + \sum_{i=1}^{d-1}(\omega_i - 1)^2[1 + 10\sin^2(\pi\omega_i + 1)] + (\omega_d - 1)^2[1 + \sin^2(2\pi\omega_d)]$	$[-10,10]$	d	0
Levy N.13	$f_{35}(x) = \sin^2(3\pi x_1) + (x_1 - 1)^2[1 + \sin^2(3\pi x_2)] + (x_2 - 1)^2[1 + \sin^2(2\pi x_2)]$	$[-10,10]$	2	0
McCormick	$f_{36}(x) = \sin(x_1 + x_2) + (x_1 - x_2)^2 - 1.5x_1 + 2.5x_2 + 1$	$[-3,4]$	2	-1.9133
Michalewicz	$f_{37}(x) = -\sum_{i=1}^{d}\sin(x_i)\sin^{2m}\left(\frac{ix_i^2}{\pi}\right)$	$[0, \pi]$	2	-1.8013
Powell	$f_{38}(x) = \sum_{i=1}^{\frac{d}{4}}[(x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} + x_{4i})^2 + (x_{4i-2} + 2x_{4i-1})^4 + 10(x_{4i-3} + x_{4i})^4]$	$[-4,5]$	d	0
Rastrigin	$f_{39}(x) = 10d + \sum_{i=1}^{d}[x_i^2 - 10\cos(2\pi x_i)]$	$[-5.12, 5.12]$	d	0
Rosenbrock	$f_{40}(x) = \sum_{i=1}^{d-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-5, 10]$	d	0
Schwefel	$f_{41}(x) = 418.9829d - \sum_{i=1}^{d}x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	$[-500,500]$	d	0
Shekel	$f_{42}(x) = -\sum_{i=1}^{10}\left(\sum_{j=1}^{4}(x_j - c_{ji})^2 + \beta_i\right)^{-1}$ where β, c are from [61]	$[0,10]$	4	-10.5364
Shubert	$f_{43}(x) = \left(\sum_{i=1}^{5} i\cos((i+1)x_1 + i)\right)\left(\sum_{i=1}^{5} i\cos((i+1)x_2 + i)\right)$	$[-10,10]$	2	-186.7309
Styblinski Tang	$f_{44}(x) = \frac{1}{2}\sum_{i=1}^{d}(x_i^4 - 16x_i^2 + 5x_i)$	$[-5,5]$	d	-39.1659d
Trid	$f_{45}(x) = \sum_{i=1}^{d}(x_i - 1)^2 - \sum_{i=2}^{d}x_ix_{i-1}$	$[d^2, d^2]$	6	-50

The settings of competing algorithms are considered similar to the presented algorithms. This helps the experiments to be fair. The number of the initial population is considered 20 for all algorithms. For the GA algorithm and the algorithms created from the combination of GA namely GAPSO and PSOGA, 70% of the population is

considered for crossover operation, and 30% of the population is considered for mutation. The type of crossover and mutation in the standard GA algorithm and its combinations are arithmetic crossover and Gaussian mutation. Table 3 shows the settings of all algorithms. The settings of all the competing algorithms are manually tuned to provide the best solutions. This means that we ran the algorithms several times with changes in the selected parameter values to achieve the best tuning conditions for the parameters. The stopping condition of the algorithms and their evaluation criterion is the Number of Function Evaluations (NFE). The default dimension for all d-dimensional benchmark test functions was considered to be ten. In experiments, after 10,000 NFE calls, the algorithm is stopped and the results are recorded.

Table 3. The values used to adjust the parameters of the algorithms

Algorithm	Parameters	Values
GA	Population size	20
	Crossover percentage	0.7
	Mutation percentage	0.3
PSO	Swarm size	20
	Inertia weight	1
	Inertia weight damping ratio	0.99
	Personal learning coefficient	2
	Global learning coefficient	2
GAPSO	Population size	20
	Other parameters are similar to GA and PSO	
PSOGA	Swarm size	20
	Other parameters are similar to PSO and GA	
GPC	Population size	20
	Gravity	9.8
	Angle of ramp	14
	Initial velocity	rand(0, 1)
	Minimum Friction	1
	Maximum Friction	5
	Substitution Probability	0.9
PSOGPC	Swarm size	20
	Other parameters are similar to PSO and GPC	
GPCPSO	Population size	20
	Other parameters are similar to GPC and PSO	

For comparison, the mean of the best solution obtained from 50 independent runs of each algorithm is considered. Table 4 and Table 5 show the mean and standard deviation obtained from 50 independent runs on unimodal and multimodal benchmark test functions. These tables show that the performance of the proposed combined algorithms namely PSOGPC and GPCPSO is acceptable and desirable. Although the results show that the GPC algorithm alone performs well, it is better when combined with PSO.

Table 4. Mean and standard deviation obtained from unimodal benchmark test functions

Fn	Algorithms						
	GA	PSO	GAPSO	PSOGA	GPC	**PSOGPC**	**GPCPSO**
f_1	0.0020±0.0084	**0.0000±0.0000**	**0.0000±0.0000**	**0.0000±0.0000**	**0.0000±0.0000**	**0.0000±0.0000**	**0.0000±0.0000**
f_2	0.0003±0.0005	**0.0000±0.0000**	**0.0000±0.0000**	**0.0000±0.0000**	**0.0000±0.0000**	**0.0000±0.0000**	**0.0000±0.0000**
f_3	-0.8398±0.3702	**-1.0000±0.0000**	**-1.0000±0.0000**	**-1.0000±0.0000**	**-1.0000±0.0000**	**-1.0000±0.0000**	**-1.0000±0.0000**
f_4	**-0.8690±0.0000**	-0.8283±0.1089	**-0.8690±0.0000**	**-0.8690±0.0000**	**-0.8690±0.0000**	**-0.8690±0.0000**	**-0.8690±0.0000**
f_5	0.1318±0.0538	0.0839±0.0384	0.0750±0.0386	0.0833±0.0360	**0.0000 ±0.0000**	**0.0000 ±0.0000**	**0.0000 ±0.0000**
f_6	0.1792±0.2108	3.66e-20±1.14e-19	2.49e-20±5.20e-20	2.23e-20±4.50e-20	9.02e-27±5.21e-26	5.78e-26±3.88e-25	**4.27e-30±1.73e-29**
f_7	0.0002±0.0005	4.47e-40±1.80e-39	**1.00e-55±4.00e-55**	1.78e-55±7.75e-55	1.08e-32±4.66e-32	3.26e-34±1.88e-33	1.02e-41±5.78e-41
f_8	47.2947±53.4984	8.1130±9.9572	7.5975±9.3436	7.5687±8.2561	**5.2521±8.2703**	7.2023±7.9863	7.4222±9.3967
f_9	0.1747±0.5308	0.0034±0.0066	0.0024±0.0029	0.0079±0.0025	0.0057±0.0086	**0.0014±0.0027**	0.0019±0.0125
f_{10}	0.0004±0.0009	**0.0000±0.0000**	**0.0000±0.0000**	**0.0000±0.0000**	**0.0000±0.0000**	**0.0000±0.0000**	**0.0000±0.0000**
f_{11}	0.5000±1.391e-06	**0.5000±2.54e-08**	0.5000±1.3524e-07	0.5000±8.87e-08	0.5000±1.00e-07	0.5000±5.81e-08	0.5000±1.45e-07
f_{12}	0.0002±0.0002	2.67e-23±5.11e-23	3.80e-23±6.33e-23	7.82e-23±1.80e-22	2.82e-29±1.34e-28	1.46e-29±5.10e-29	**5.16e-33±1.97e-32**
f_{13}	2.37e-07±5.72e-07	2.96e-39±1.74e-38	4.73e-45±2.48e-44	2.74e-45±1.55e-44	1.95e-34±9.68e-34	3.73e-41±2.04e-40	**9.35e-47±3.26e-46**
f_{14}	0.0039±0.0057	9.71e-22±2.98e-21	4.21e-22±8.57e-22	5.99e-22±1.23e-21	3.51e-27±2.20e-26	1.35e-28±7.76e-28	**8.24e-32±3.80e-31**

f_{15}	6.2082±5.6592	1.26e-11±6.14e-11	2.45e-12±8.98e-12	5.86e-13±1.97e-12	6.03e-28±3.38e-27	5.25e-27±2.78e-26	**1.22e-29±3.99e-29**

Table 5. Mean and standard deviation obtained from multimodal benchmark test functions

Fn	Algorithms						
	GA	PSO	GAPSO	PSOGA	GPC	**PSOGPC**	**GPCPSO**
f_{16}	0.1158±0.0839	0.1155±0.3500	7.30e-08±3.90e-07	3.69e-08±1.63e-07	2.30e-14±6.65e-14	1.60e-14±4.48e-14	**1.77e-15±1.90e-15**
f_{17}	0.1429±0.3524	0.1524±0.3079	0.1914±0.2501	0.1524±0.3079	0.0457±0.1828	0.1177±0.4163	**3.96e-19±2.53e-18**
f_{18}	**0.3978±0.0000**	**0.3978±0.0000**	0.4457±0.3384	**0.3978±0.0000**	**0.3978±0.0000**	**0.3978±0.0000**	**0.3978±0.0000**
f_{19}	0.1061±0.0242	0.1007±0.0097	0.0977±0.0120	0.1023±0.0117	**0.1000±0.0000**	**0.1000±0.0000**	**0.1000±0.0000**
f_{20}	**-1.0316±0.0000**	**-1.0316±0.0000**	**-1.0316±0.0000**	**-1.0316±0.0000**	**-1.0316±0.0000**	**-1.0316±0.0000**	**-1.0316±0.0000**
f_{21}	1.74e-06±4.82e-06	1.45e-44±5.79e-44	4.19e-63±1.12e-62	5.82e-63±1.75e-62	3.79e-33±1.46e-32	**0.0000±0.0000**	3.21e-60±2.27e-59
f_{22}	1.8884±3.1227	0.3916±1.5464	**7.29e-07±1.45e-06**	1.10e-06±2.15e-06	0.7813±1.3650	0.0019±0.0025	0.3733±0.8898
f_{23}	**-2.0626±0.0000**	**-2.0626±0.0000**	**-2.0626±0.0000**	**-2.0626±0.0000**	**-2.0626±0.0000**	**-2.0626±0.0000**	**-2.0626±0.0000**
f_{24}	0.9990±3.94e-10	5.9646±3.2557	**0.9980±0.0000**	**0.9980±0.0000**	5.8602±3.8533	**0.9980±0.0000**	**0.9980±0.0000**
f_{25}	0.7428±0.3830	0.6133±0.1827	0.6000±0.2020	0.6133±0.1827	0.6667±6.63e-05	**0.5733±0.2336**	0.6666±2.92e-16
f_{26}	-0.9693±0.0321	-0.9859±0.0266	-0.9783±0.0305	-0.9847±0.0275	**-1.0000±0.0000**	**-1.0000±0.0000**	**-1.0000±0.0000**
f_{27}	-846.560±97.572	-774.296±124.591	-801.637±135.094	-850.525±121.583	**-956.866±4.485**	-953.531±19.995	-956.754±13.986
f_{28}	**-6.0207±0.0000**	**-6.0207±0.0000**	**-6.0207±0.0000**	**-6.0207±0.0000**	**-6.0207±0.0000**	**-6.0207±0.0000**	**-6.0207±0.0000**
f_{29}	**-3.8628±0.0000**	-3.8164±0.1854	-3.8319±0.1530	**-3.8628±0.0000**	**-3.8628±0.0000**	**-3.8628±0.0000**	**-3.8628±0.0000**
f_{30}	-3.0964±0.0881	-3.0440±0.1167	-3.0583±0.1121	-3.0583±0.1121	-3.0155±0.1202	-3.0985±0.0842	**-3.1345±2.88e-13**
f_{31}	-3.0289±0.0257	-3.0253±0.0278	-3.0253±0.0278	-3.0240±0.0284	-3.0088±0.0342	-3.0240±0.0284	**-3.0302±0.0248**
f_{32}	**-19.2085±0.0000**	-17.5244±3.6417	**-19.2085±0.0000**	**-19.2085±0.0000**	**-19.2085±0.0000**	**-19.2085±0.0000**	**-19.2085±0.0000**
f_{33}	-3.7693±0.7035	-3.4805±0.9214	-3.6554±0.8304	-3.4213±0.9315	-3.6090±0.87194	-3.7933±0.1361	**-3.9827±0.1967**
f_{34}	0.0003±0.0003	0.3831±0.7424	**4.76e-21±1.27e-20**	3.29e-20±1.89e-19	0.7182±0.0354	0.4905±0.1285	0.1898±0.1349
f_{35}	1.49e-05±4.46e-05	1.34e-30±1.76e-46	1.34e-30±1.76e-46	1.34e-30±1.76e-46	4.01e-08±9.11e-08	7.09e-28±1.83e-27	**1.72e-31±2.68e-31**
f_{36}	**-1.9132±0.0000**	-1.9112±0.0144	**-1.9132±0.0000**	**-1.9132±0.0000**	**-1.9132±0.0000**	**-1.9132±0.0000**	**-1.9132±0.0000**
f_{37}	**-1.8013±0.0000**	**-1.8013±0.0000**	**-1.8013±0.0000**	**-1.8013±0.0000**	**-1.8013±0.0000**	**-1.8013±0.0000**	**-1.8013±0.0000**
f_{38}	0.0934±0.0699	1.12e-05±9.26e-06	9.20e-07±7.88e-07	8.33e-07±7.04e-07	7.36e-29±2.38e-28	**1.18e-31±7.51e-31**	2.02e-31±1.36e-30
f_{39}	0.0285±0.0305	12.6161±5.6514	0.2392±0.5533	0.1867±0.3839	**0.0000±0.0000**	**0.0000±0.0000**	**0.0000±0.0000**
f_{40}	17.5958±25.4988	15.8734±12.0997	10.5263±0.5232	10.5945±0.6978	8.4080±0.1127	**3.4455±9.9025**	5.1537±0.2881
f_{41}	1511.914±200.840	1661.773±310.272	1553.896±169.592	1572.057±198.981	1472.102±349.439	1227.854±269.592	**1165.091±327.604**
f_{42}	-5.5208±3.4998	-5.3482±3.0348	-5.7969±3.6541	-5.4833±3.4152	-5.5657±3.2387	-6.1105±3.4255	**-6.3360±3.6849**
f_{43}	-186.723±0.0241	**-186.730±0.0000**	**-186.730±0.0000**	**-186.730±0.0000**	**-186.730±0.0000**	**-186.730±0.0000**	**-186.730±0.0000**
f_{44}	-33.462±1.976	-34.642±2.098	-34.670±1.799	-34.642±1.783	-33.963±2.310	-34.716±1.778	**-36.310±2.562**
f_{45}	-48.9673±0.7960	**-50.0000±0.0000**	**-50.0000±0.0000**	**-50.0000±0.0000**	**-50.0000±0.0000**	**-50.0000±0.0000**	**-50.0000±0.0000**

Bohachevsky (f_1) and Booth (f_2) functions are both convex, unimodal functions and are defined for two-dimensional space. For these two functions, all algorithms except GA provide the best solution. In the case of the Easom (f_3) function, which is a unimodal function defined for two-dimensional space, all algorithms provide the best solution. Of course, only GA could not find the best solution for this function. The Gramacy & Lee (f_4) function is a simple one-dimensional function. This function is also unimodal. For this simple function, all competing algorithms provide the best solution except PSO. Griewank (f_5), Schaffer N.2 (f_{10}) and Schaffer N.4 (f_{11}) functions are non-convex, unimodal, differentiable, and non-separable functions. Schaffer N.2 and Schaffer N.4 functions are defined for two-dimensional space, but the Griewank function covers d-dimensional space. For the Griewank function, the best solutions are related to GPC, PSOGPC, and GPCPSO algorithms. For Schaffer function N.2, all algorithms except GA found the best solution. For the Schaffer function, N.4 all the solutions given by the algorithms are good, but if we consider the standard deviation of the runs, PSO has performed slightly better. Matyas (f_7) function is an almost simple function. This function is convex, unimodal, differentiable, and non-separable. This function is also defined for two-dimensional space. For this function, GAPSO has provided better performance than other algorithms. After that GPCPSO performed better. Sphere (f_{12}), Sum-Powers (f_{13}) and Sum-Squares (f_{14}) functions are also almost simple functions. These functions are convex, unimodal, and separable. The GPCPSO algorithm provides the best solutions for these three functions. The three functions namely Hyper-Ellipsoid (f_6), Perm (f_8), and Zakharov (f_{15}) are unimodal and defined for d-dimensional space. The GPCPSO algorithm provides the best solutions for Hyper-Ellipsoid and Zakharov functions. In the case of Perm function, the best solution is related to GPC. The Power Sum function (f_9) also is a unimodal function that can be defined for four-dimensional space. PSOGPC provides the best solution for this function.

Camel Six-Hump (f_{20}), Cross-In-Tray (f_{23}), Forrester (f_{28}) and Michalewicz (f_{37}) functions are multimodal, non-convex, and non-separable functions. For these four functions, all competing algorithms have similar solutions

and were able to provide the best solutions based on our experiments. Ackley (f_{16}), Beale (f_{17}), Camel Three-Hump (f_{21}), Levy (f_{34}), Levy N.13 (f_{35}), and Shubert (f_{43}) functions are all non-convex, multimodal, differentiable, and non-separable functions. These functions are considered as complex functions. For the Ackley, Beale, and Levy N.13 function, the best solution is given by the proposed GPCPSO. For the Camel Three-Hump function, the best solution is related to PSOGPC. For the Levy function, the GAPSO algorithm has provided the best solution by a significant difference. In the case of the Shubert function, all algorithms except GA have reached the best solution. Branin (f_{18}) function is a multimodal function defined for two-dimensional space. All algorithms except GAPSO perform well for this function. In the case of the Bukin (f_{19}) function, which is continuous, multimodal, non-differentiable, non-separable, and is defined for two-dimensional space, GPC, PSOGPC, and GPCPSO algorithms provide the best solution. The Colville (f_{22}) function is defined for four-dimensional space. The best solution for this function is GAPSO followed by PSOGPC. The three functions Dixon-Price (f_{25}), Rastrigin (f_{39}), and Rosenbrock (f_{40}) are hard, convex, multimodal, differentiable, and defined for n-dimensional space. The best solution for Dixon-Price and Rosenbrock is provided by PSOGPC. Regarding the Rastrigin function, three algorithms namely GPC, PSOGPC, and GPCPSO perform better than other algorithms.

Other functions, such as De Jong (f_{24}), Drop-Wave (f_{26}), and Powell (f_{38}), are all multimodal and non-convex. For the De Jong function, all the combined algorithms in the experiments provided the best performance. For the Drop-Wave function, three algorithms namely GPC, PSOGPC, and GPCPSO were better than other algorithms. As for the Powell function, the best performance is related to PSOGPC. The three functions Hartmann-3D (f_{29}), Hartmann-4D (f_{30}), and Hartmann-6D (f_{31}) are multimodal and are defined for three, four, and six-dimensional space, respectively. For Hartmann-3D, except for PSO and GAPSO algorithms, the remaining algorithms provide the best solutions. In the case of Hartmann-4D and Hartmann-6D, the GPCPSO algorithm was the best among other algorithms. The Holder-Table (f_{32}) function is also a non-convex, multimodal, non-differentiable, and non-separable function. For this function, except for PSO, the rest of the algorithms found the best solution. For the McCormick (f_{36}) function, only PSO does not provide a desirable solution. The Langermann (f_{33}) function is defined for two-dimensional space. For this function, GPCPSO has been better than other algorithms. For the Eggholder (f_{27}) function, which is a hard function to optimize due to having large local optimal points, the GPC algorithm provided the best solution. The Schwefel (f_{41}) function is a function for the d-dimensional space. This function has a large search space. The GPCPSO algorithm provides the best solution for this function. For the Shekel (f_{42}) and Styblinski Tang (f_{44}) functions, the GPCPSO algorithm provides the best solution. The Trid function (f_{45}) is multimodal, bowl-shaped, and defined for six-dimensional space. For this function, except for GA, the rest of the algorithms provided desirable solutions. In general, the results of the benchmark functions show that the presented hybrid algorithms perform better than other competing algorithms in most functions.

One of the strengths of the GPC algorithm is its memory. This means that previous good information is retained. One of the strengths of PSO is search directionality, meaning that population members are constantly biased toward the best search agent. The combination of these two features, namely memory and leaning towards a better search factor in the proposed combined algorithms, has caused the performance to improve to some extent when facing some complex functions. One of the problems of PSO is premature convergence. The combination of PSO with GPC has completely removed this weakness.

5 Statistical analysis

Statistical analysis is used to discover the meaning of the data. For this purpose, relationships and probabilities between data are determined quantitatively. The main goal is to identify trends. In this paper, Friedman and Iman-Davenport tests have been used to find significant differences between the results obtained from the two presented hybrid algorithms, namely PSOGPC and GPCPSO, and other competing algorithms. Table 6 shows Friedman's ranking based on the results obtained from Table 4 and Table 5.

Table 6. Ranking of the algorithms

	Algorithm						
	GA	PSO	GAPSO	PSOGA	GPC	**PSOGPC**	**GPCPSO**
Ranking	5.56	5.06	3.87	4.09	3.93	**2.96**	**2.54**

Table 6 shows that the best rank among all the algorithms used in the experiments is related to the GPCPSO algorithm. After the GPCPSO, there are PSOGPC, GAPSO, GPC, PSOGA, PSO, and GA, respectively. Interestingly, the combinations created with GPC and PSO rank better than the combinations created with GA and PSO. It seems that it can be concluded that the GPC mechanism is better than the GA mechanism to improve PSO.

In the comparison between GPC, PSO, and GA, GPC ranks better than the other two algorithms. Table 7 shows the results of the Friedman and Iman-Davenport tests.

Table 7. Results of Friedman's and Iman–Davenport's tests

Test method	Chi-Square	Degrees of freedom (DF)	*p*-Value	Hypothesis
Friedman	89.7480	6	3.417e-17	Rejected
Iman–Davenport	14.0410	6	7.261e-14	Rejected

Table 7 shows hypothesis is rejected. In this way, it can be concluded that there is a significant difference between the performance of the algorithms. To prove the existence of a significant difference, post-hoc tests can be used. This helps in better analysis. In this paper, Holm's method is used as a post-hoc test for better analysis. This test compares the best rank obtained from the Friedman ranking one by one with the results of other algorithms. As Table 6 showed, the best ranking is related to the GPCPSO algorithm. So this algorithm is considered as a control algorithm and is compared one by one with other competing algorithms. Also, consider that the confidence interval is 95% ($\alpha = 0.05$). The results obtained from Holm's method are shown in Table 8.

Table 8. Results obtained from Holm's method as a post-hoc test

Algorithm	*j*	α/*j*	*z*-Score	*p*-Value	Hypothesis
PSOGPC	1	0.05000	0.92222	0.356424	Not rejected
GAPSO	2	0.02500	2.92038	0.003497	Rejected
GPC	3	0.01666	3.05212	0.002272	Rejected
PSOGA	4	0.01250	3.40345	0.000666	Rejected
PSO	5	0.01000	5.53335	< 0.00001	Rejected
GA	6	0.00833	6.63124	< 0.00001	Rejected

The results show that the combinations created by GPC and PSO algorithms are not significantly different. Although GPCPSO is slightly better than PSOGPC according to experiments and according to Table 8, overall they are not significantly different. But according to Table 8, GPCPSO is significantly different from other competing algorithms.

6 Solving classical engineering problems

Engineering problems may have multiple equality and inequality constraints, so a constraint management method is required to solve constrained problems optimally. The main challenge of managing the constraint is the direct effects of the fitness function on the position of the search agents. The essence of the GPC algorithm and its hybrid derivatives is that there is no direct relationship between the search factors and the fitness function. Therefore, by creating appropriate constraints and penalty functions, the constraints can be managed and dealt with. In this way, in GPC, PSOGPC, and GPCPSO, if any worker violates the constraints, they can be substituted by another worker in the next iteration. In this paper, to solve engineering problems, it is tried to use simple and scalar penalty functions. A comparison of the results with competing algorithms has been done and presented in appropriate tables. It is worth noting that for this section, two recently published powerful algorithms have also been included in the experiments. These two algorithms are Golden Jackal Optimization (GJO) [62] and White Shark Optimizer (WSO) [63]. Note that the experiment conditions are similar for all algorithms to make a fair comparison. Also, the settings of the parameters of all algorithms participating in the comparisons have been done based on Table 3, with the difference that each algorithm has been run 30 times independently and the best, mean, and standard deviation of runs have been reported. In the case of GJO, the parameter C_1 is considered to be 1.5, and for WSO the maximum wavy motion parameter is 0.75 and the minimum wavy motion is 0.07.

6.1 Gear train design problem

The gear train problem is discrete. The objective is to minimize the gear ratio by finding the optimal number of teeth for the four gears of the train. To handle discrete parameters, each search factor is rounded to the nearest integer before evaluating the fitness function. Figure 6 shows the parameters and variables of the problem. The parameters are the number of teeth of the gears. There are a total of four variables for this problem. Also, the problem is unconstrained, although the range of variables is considered a constraint. Consider that $\vec{x} = [x_1\ x_2\ x_3\ x_4] = [n_A n_B n_C n_D]$, the following function should be minimized.

$$f(\vec{x}) = \left(\frac{1}{6.931} - \frac{x_3 x_2}{x_1 x_4}\right)^2 \tag{7}$$

Also, the range of changes for variables is $12 \leq x_1, x_2, x_3, x_4 \leq 60$. Table 9 shows the results of running the algorithms for this problem. Figure 7 shows the convergence diagram for 40 iterations of the best solution for some of the competing algorithms. For this problem, PSOGPC has performed better than other algorithms. Then the mean of the best performance is related to GPC.

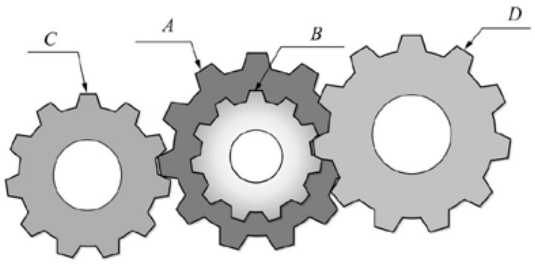

Fig 6. Parameters and variables of the gear train design problem

Table 9. The results obtained from the algorithms in solving the gear train design problem

Algorithm	Optimum variables				Cost		
	n_A	n_B	n_C	n_D	Best	Mean	Std
GA	53	13	30	51	2.307e-11	1.808e-07	3.345e-07
PSO	43	16	19	49	2.700e-12	3.163e-09	5.267e-09
GAPSO	51	26	15	53	2.307e-11	7.620e-09	1.090e-08
PSOGA	49	16	19	43	2.700e-12	6.014e-09	1.177e-08
GJO	49	16	19	43	1.391e-12	4.482e-08	4.443e-08
WSO	33	14	17	50	1.101e-09	2.751e-08	1.171e-07
GPC	49	19	16	43	2.700e-12	2.017e-09	2.355e-09
PSOGPC	49	16	19	43	2.700e-12	1.779e-09	5.046e-09
GPCPSO	53	13	30	51	2.307e-11	2.790e-09	3.887e-09

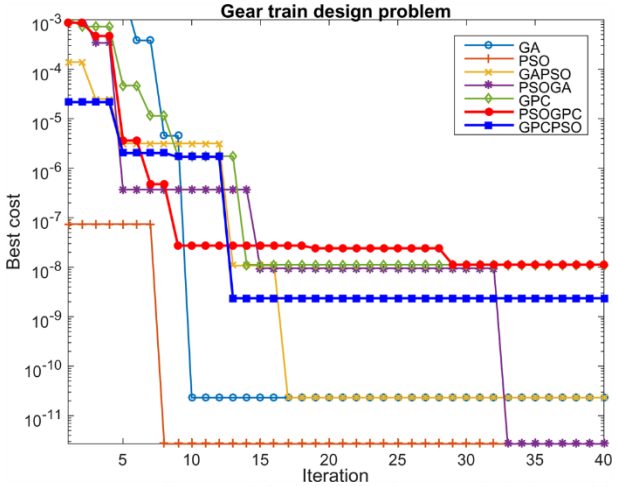

Fig 7. Convergence diagram of algorithms for the gear train design problem

6.2 Pressure vessel design problem

A pressurized vessel is a closed container designed to store fluid at a pressure different from the ambient pressure (atmosphere). Pressure difference is a dangerous parameter and due to changes in this parameter in pressure vessel, there is a possibility of explosion and destruction. As a result, the optimal design and construction of these vessels is very important. The main objective in this paper is to minimize the total cost consisting of materials, forming, and welding of a pressure vessel. Figure 8 shows the parameters and variables of the problem. In this problem, there are four variables, which are shell thickness (T_s), head thickness (T_h), inner radius (R), and the length of the cylinder without considering the head (L). Consider that $\vec{x} = [x_1\ x_2\ x_3\ x_4] = [T_s T_h R L]$, the following function should be minimized,

$$f(\vec{x}) = 0.6224 x_1 x_2 x_3 + 1.7781 x_2 x_3^2 + 3.1661 x_1^2 x_4 + 19.84 x_1^2 x_3 \quad (8)$$

subject to,

$$\begin{cases} g_1(\vec{x}) = -x_1 + 0.0193x_3 \le 0 \\ g_2(\vec{x}) = -x_3 + 0.00954x_3 \le 0 \\ g_3(\vec{x}) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \le 0 \\ g_4(\vec{x}) = x_4 - 240 \le 0 \end{cases} \quad (9)$$

Also, the range of changes for variables is $0 \le x_1 \le 99$, $0 \le x_2 \le 99$, $10 \le x_3 \le 200$, and $10 \le x_4 \le 200$. This problem has been solved in many kinds of research for optimization. The results of solving this problem in this paper are reported in Table 10. The results show that both PSOGPC and GPCPSO algorithms have been successful in providing the most optimal cost function solution. Although PSOGPC has been slightly better than GPCPSO. Figure 9 shows the convergence diagram of the best run for 200 iterations for some of the competing algorithms.

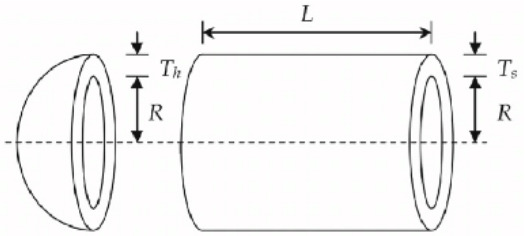

Fig 8. Parameters and variables of the pressure vessel design problem

Table 10. The results obtained from the algorithms in solving the pressure vessel design problem

Algorithm	Optimum variables				Cost		
	T_s	T_h	R	L	Best	Mean	Std
GA	0.87411	0.41619	43.62575	158.6844	6219.9757	7022.8512	509.2740
PSO	0.78137	0.38623	40.48577	197.7000	6090.8366	6394.2436	248.8347
GAPSO	0.82633	0.40849	42.81486	167.9579	6173.0758	6456.2474	326.7510
PSOGA	0.84554	0.41797	43.81068	156.5153	6010.8526	6366.3486	364.7259
GJO	0.78639	0.41539	40.69469	195.4048	5996.9671	6698.4995	534.2963
WSO	0.77922	0.38478	40.31968	199.9798	6085.3692	6206.0570	249.9967
GPC	0.82115	0.40616	42.48257	173.9046	6018.8532	6535.5043	423.2782
PSOGPC	0.80907	0.39992	41.92086	178.8500	5940.3298	6196.5115	266.3042
GPCPSO	0.83779	0.41412	43.40892	161.0474	5995.2572	6350.634	290.2151

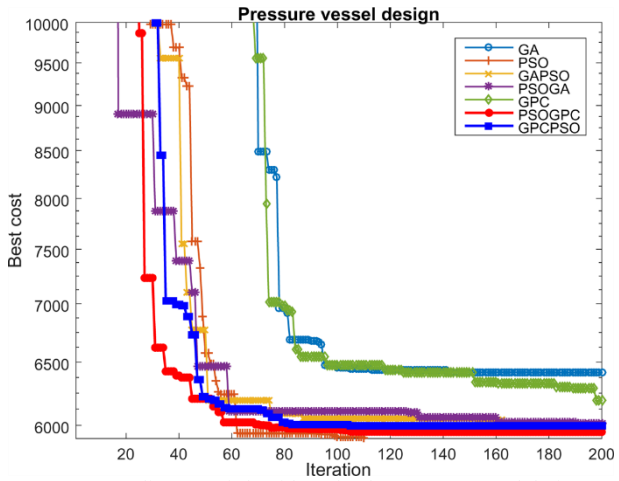

Fig 9. Convergence diagram of algorithms for the pressure vessel design problem

6.3 Speed reducer design problem

A speed reducer is part of the gearbox of a mechanical system, and of course, it may be used in many other types of applications. The speed reducer design problem is an optimization problem and considering that it has seven design variables, it creates many challenges. The goal of this problem is to minimize the weight of the speed reducer according to the constraints. Among the constraints, we can mention the bending stress of the gear teeth, surface stress, transverse deviation of the shaft, and stresses in the shafts. The variables of this problem are the width of the face (b), the module of the teeth (m), the number of teeth in the pinion (z), the length of the first shaft between the bearings (l_1), the length of the second shaft between the bearings (l_2), the diameter of the first shaft (d_1) and the

diameter of the second shaft (d_2). Figure 10 shows the parameters and variables of the problem. Consider that $\vec{x} = [x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7] = [bmzl_1l_2d_1d_2]$, the following function should be minimized,

$$f(\vec{x}) = 0.7854x_1x_2^2(3.33x_3^2 + 14.9334x_3 - 43.0934 - 1508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_4^3) + 0.7854(x_4x_6^2 + x_5x_7^2) \tag{10}$$

subject to,

$$\begin{cases} g_1(\vec{x}) = \dfrac{27}{x_1x_2^2x_3} - 1 \le 0 \\ g_2(\vec{x}) = \dfrac{397.5}{x_1x_2^2x_3^2} - 1 \le 0 \\ g_3(\vec{x}) = \dfrac{1.93x_4^3}{x_2x_6^4x_3} - 1 \le 0 \\ g_4(\vec{x}) = \dfrac{1.93x_5^3}{x_2x_7^4x_3} - 1 \le 0 \\ g_5(\vec{x}) = \dfrac{\left[\left(745\frac{x_4}{x_2x_3}\right)^2 + 16.9 \times 10^6\right]^{\frac{1}{2}}}{110x_6^3} - 1 \le 0 \\ g_6(\vec{x}) = \dfrac{\left[\left(745\frac{x_5}{x_2x_3}\right)^2 + 157.5 \times 10^6\right]^{\frac{1}{2}}}{85x_7^3} - 1 \le 0 \\ g_7(\vec{x}) = \dfrac{x_2x_3}{40} - 1 \le 0 \\ g_8(\vec{x}) = \dfrac{5x_2}{x_1} - 1 \le 0 \\ g_9(\vec{x}) = \dfrac{x_1}{12x_2} - 1 \le 0 \\ g_{10}(\vec{x}) = \dfrac{1.5x_6 + 1.9}{x_4} - 1 \le 0 \\ g_{11}(\vec{x}) = \dfrac{1.1x_7 + 1.9}{x_5} - 1 \le 0 \end{cases} \tag{11}$$

Also, the range of changes for variables is $2.6 \le x_1 \le 3.6$, $0.7 \le x_2 \le 0.8$, $17 \le x_3 \le 28$, $7.3 \le x_4 \le 8.3$, $7.3 \le x_5 \le 8.3$, $2.9 \le x_6 \le 3.9$, and $5 \le x_7 \le 5.5$. The results of solving this problem in this paper are reported in Table 11. The convergence diagram for 200 iterations of the best solution is also presented in Figure 11 for some of the competing algorithms. If we consider the mean of 30 independent runs, PSOGPC provides the best solution, followed by the PSO algorithm. But the best solution is given by GPCPSO.

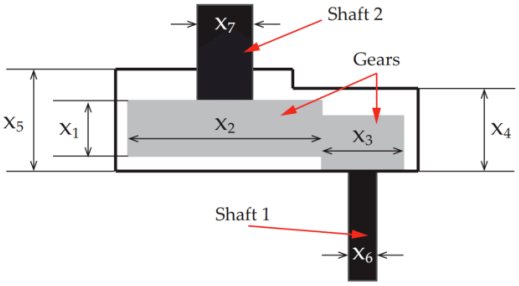

Fig 10. Parameters and variables of the speed reducer design problem

Table 11. The results obtained from the algorithms in solving the speed reducer design problem

Algorithm	Optimum variables							Cost		
	b	m	z	l_1	l_2	d_1	d_2	Best	Mean	Std
GA	4.8741	0.7037	12.8348	6.9860	7.7612	3.3632	5.3131	3072.1726	3110.1438	24.1561
PSO	3.5000	0.7000	17.0000	7.3000	7.7153	3.3502	5.2866	2994.4781	2997.8044	18.2574
GAPSO	3.5500	0.7010	15.5219	7.7047	7.9191	3.3610	5.2890	3081.4164	3123.5551	34.5377
PSOGA	3.5430	0.7039	15.4173	8.0178	7.7535	3.3717	5.2980	3081.1945	3110.4345	28.5357
GJO	3.5016	0.7000	17.0000	7.3301	8.1155	3.3508	5.2949	3009.4687	3030.5225	10.2694
WSO	3.5002	0.7000	17.0000	7.3001	7.7161	3.3503	5.2869	2994.6621	3013.9959	23.4243
GPC	3.5015	0.7000	17.0000	7.3000	8.3000	3.3537	5.2910	3011.5986	3022.2354	8.57380
PSOGPC	3.5000	0.7000	17.0000	7.3000	7.7153	3.3502	5.2866	2994.4719	2994.6103	0.69200
GPCPSO	3.5000	0.7000	17.0000	7.3000	7.7153	3.3502	5.2866	2994.4716	2998.2355	4.58540

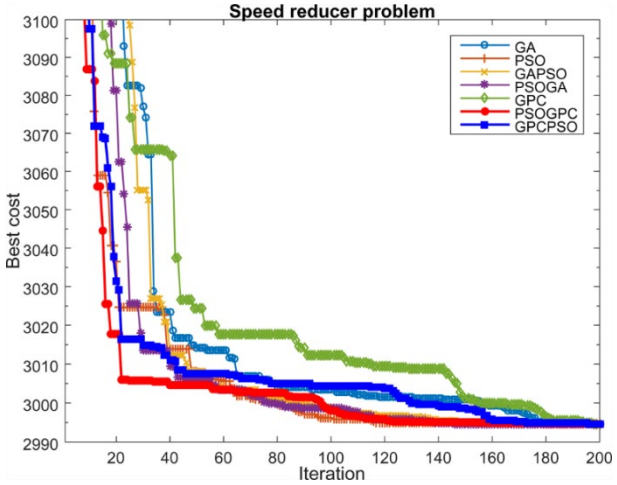

Fig 11. Convergence diagram of algorithms for the speed reducer design problem

6.4 Tension/compression spring design problem

In this problem, the weight of the tension/compression spring should be minimized. This problem is included in the category of continuous constrained optimization problems. Figure 12 shows the parameters and variables of the tension/compression spring problem. The wire diameter (d), the average coil diameter (D) and the number of active coils (N) are the variables of the problem. Consider that $\vec{x} = [x_1\ x_2\ x_3] = [dDN]$, the following function should be minimized,

$$f(\vec{x}) = (x_3 + 2)x_2x_1^2 \tag{12}$$

subject to,

$$\begin{cases} g_1(\vec{x}) = 1 - \dfrac{x_1^2 x_3}{71785x_1^4} \leq 0 \\ g_2(\vec{x}) = \dfrac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \dfrac{1}{5108x_1^2} \leq 0 \\ g_3(\vec{x}) = 1 - \dfrac{140.45x_1}{x_2^2x_3} \leq 0 \\ g_4(\vec{x}) = \dfrac{x_1 + x_2}{1.5} - 1 \leq 0 \end{cases} \tag{13}$$

Also, the range of changes for variables is $0.05 \leq x_1 \leq 2$, $0.25 \leq x_2 \leq 1.3$, and $2 \leq x_3 \leq 15$. The results are presented in Table 12. According to the results in Table 12, if we consider the average of 30 independent executions, the PSOGPC algorithm has been better than other algorithms. Figure 13 shows the convergence diagram of the first 100 iterations of the best independent run for some of the competing algorithms.

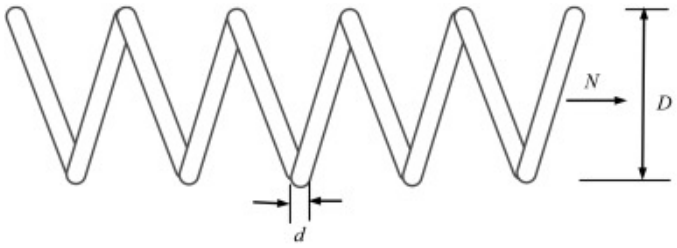

Fig 12. Parameters and variables of the tension/compression spring design problem

Table 12. The results obtained from the algorithms in solving the tension/compression spring design problem

Algorithm	Optimum variables			Cost		
	d	D	N	Best	Mean	Std
GA	0.05885	0.54278	5.40070	0.013913	0.021068	0.014549
PSO	0.05172	0.35768	11.2323	0.012665	0.013815	0.001350
GAPSO	0.05106	0.34196	12.2093	0.012672	0.013333	0.000746
PSOGA	0.05169	0.35691	11.2776	0.012665	0.013225	0.000601
GJO	0.05000	0.31719	14.1845	0.012753	0.012866	0.000212
WSO	0.05169	0.35666	11.2931	0.012679	0.012689	5.544e-05
GPC	0.05277	0.38313	9.89860	0.012695	0.014786	0.002238
PSOGPC	0.05148	0.35187	11.5784	0.012666	0.012724	7.996e-05
GPCPSO	0.05323	0.39506	9.35060	0.012708	0.014623	0.001685

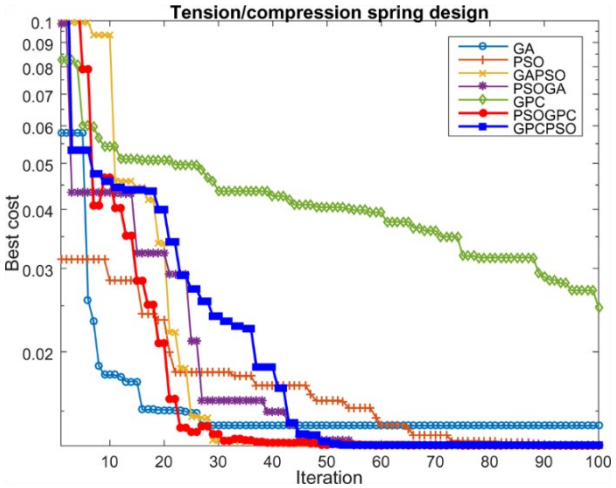

Fig 13. Convergence diagram of algorithms for the tension/compression spring design problem

6.5 Three-bar truss design problem

Truss is a multi-member structure, all its parts are pinned together. Pin means that there is no torque in any joint. Therefore, there is only force in the truss. In this paper, the design of the three-bar truss is considered. The goal is to minimize the weight of the truss. The main challenge is the issue of limited search space and its difficulty. Constraints of the problem include stress, deflection, and buckling. Figure 14 shows the parameters and variables of the problem. Consider that $\vec{x} = [x_1\ x_2] = [A_1 A_2]$, the following function should be minimized,

$$f(\vec{x}) = (2\sqrt{2}x_1 + x_2) \times l \tag{14}$$

subject to,

$$\begin{cases} g_1(\vec{x}) = \dfrac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \leq 0 \\ g_2(\vec{x}) = \dfrac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \leq 0 \\ g_2(\vec{x}) = \dfrac{1}{\sqrt{2}x_2 + x_1} P - \sigma \leq 0 \end{cases} \tag{15}$$

where $l = 100\ cm$, $P = 2\ kN/cm^2$, and $\sigma = 2\ kN/cm^2$. Also, the range of changes for variables is $0 \leq x_1, x_2 \leq 1$. Table 13 shows the results of running the algorithms for this problem. Figure 15 shows the convergence diagram for 100 iterations of the best solution for some algorithms. If we consider the mean of 30 independent runs, the GPC algorithm has performed better than the other algorithms. After that, the GPCPSO algorithm shows a better performance than the competing algorithms.

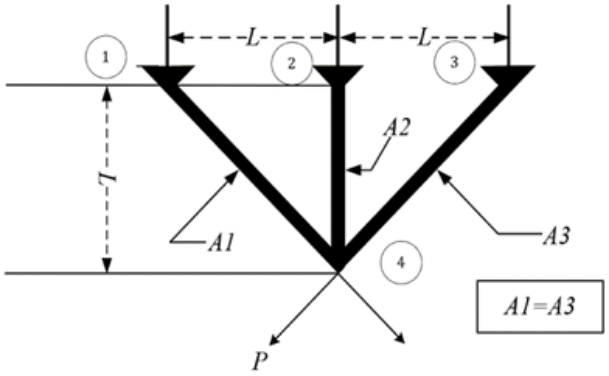

Fig 14. Parameters and variables of the three-bar truss design problem

Table 13. The results obtained from the algorithms in solving the three-bar truss design problem

Algorithm	Optimum variables		Cost		
	A_1	A_2	Best	Mean	Std
GA	0.78765	0.41117	263.8966	264.5439	1.18700
PSO	0.78875	0.40804	263.8958	263.9200	0.06712
GAPSO	0.78868	0.40824	263.8958	263.9143	0.05021
PSOGA	0.78879	0.40792	263.8959	263.9093	0.04762
GJO	0.78727	0.41315	263.9049	265.9212	1.96562
WSO	0.78871	0.40899	263.8965	263.9162	0.02256
GPC	0.78875	0.40804	263.8958	263.9000	0.00712
PSOGPC	0.78951	0.40590	263.8958	263.9139	0.03933
GPCPSO	0.78863	0.40839	263.8958	263.9045	0.01635

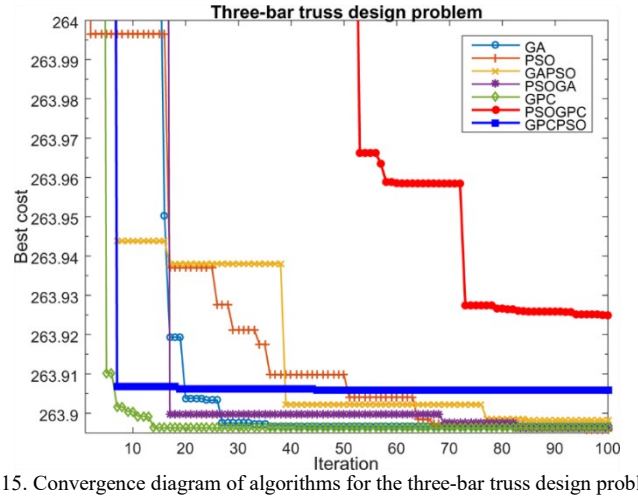

Fig 15. Convergence diagram of algorithms for the three-bar truss design problem

6.6 Welded beam design problem

In the process of analyzing a beam, many factors such as the type of structure, the type of constituent materials, the type of applied loads, environmental conditions, and construction cost are considered. However, considering the resistance factor, the shape and size of the beam should be such that the applied stresses do not exceed the allowable stresses of its constituent material. The purpose of the welding beam design problem is to reduce the fabrication cost of the welding beam in such a way that its cost should be optimized and minimized. Optimization of construction cost is influenced by shear stress (τ), bending stress in the beam (θ), buckling load on the bar (P_c), end deflection of the beam (δ), and side constraints. This problem has four variables. These variables are weld thickness (h), bar thickness (b), bar height (t), and the length of the part attached to the bar (l), which is shown in Figure 16. Consider that $\vec{x} = [x_1\ x_2\ x_3\ x_4] = [hltb]$, the following function should be minimized,

$$f(\vec{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4x_3(14 + x_2) \quad (16)$$

subject to,

$$\begin{cases} g_1(\vec{x}) = \tau(\vec{x}) - \tau_{max} \le 0 \\ g_2(\vec{x}) = \sigma(\vec{x}) - \sigma_{max} \le 0 \\ g_3(\vec{x}) = \delta(\vec{x}) - \delta_{max} \le 0 \\ g_4(\vec{x}) = x_1 - x_4 \le 0 \\ g_5(\vec{x}) = P - P_c(\vec{x}) \le 0 \\ g_6(\vec{x}) = 0.125 - x_1 \le 0 \\ g_7(\vec{x}) = 1.10471 x_1^2 + 0.04811 x_3 x_4 (14 + x_2) - 5 \le 0 \end{cases} \quad (17)$$

where $\tau(\vec{x}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}$ in such a way that $\tau' = \frac{P}{\sqrt{2}x_1x_2}$ and $\tau'' = \frac{P\left(L+\frac{x_2}{2}\right)R}{J}$, where $R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1+x_3}{2}\right)^2}$ and $J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{4} + \left(\frac{x_1+x_3}{2}\right)^2\right]\right\}$. Also, $\sigma(\vec{x}) = \frac{6PL}{x_4x_3^2}$ and $\delta(\vec{x}) = \frac{6PL^3}{Ex_3^2x_4}$. In addition, we have $P_c(\vec{x}) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)$ where $= 6000\,l$, $L = 14\,i$, $\delta_{max} = 0.25\,i$, $E = 3 \times 10^6\,ps$, $G = 12 \times 10^6\,psi$, $\tau_{max} = 13600\,ps$, and $\sigma_{max} = 30000\,psi$. The range of changes for variables is $0.1 \le x_1 \le 2$, $0.1 \le x_2 \le 10$, $0.1 \le x_3 \le 10$, and $0.1 \le x_4 \le 2$. Solving this problem has also been done in many studies. The results of solving this problem are reported in table 14. The convergence diagram for 200 iterations of the best solution is also presented in Figure 17. Among the algorithms that were analyzed, GPCPSO has a better performance than other competing algorithms. Although except for GA, the rest of the algorithms found and provided the best solution.

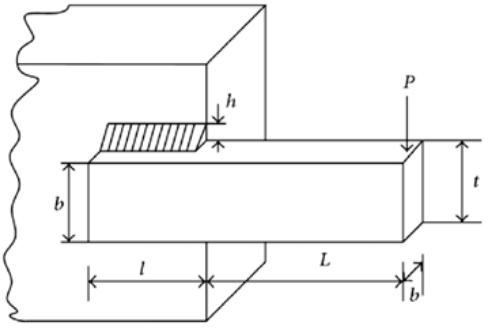

Fig 16. Parameters and variables of the welded beam design problem

Table 14. The results obtained from the algorithms in solving the welded beam design problem

Algorithm	Optimum variables				Cost		
	h	l	t	b	Best	Mean	Std
GA	0.20794	4.3912	7.1652	0.32876	2.2940	3.1065	0.5745
PSO	0.20573	3.4705	9.0366	0.20573	1.7249	1.9262	0.2863
GAPSO	0.20573	3.4705	9.0366	0.20573	1.7249	1.8667	0.2399
PSOGA	0.20573	3.4705	9.0366	0.20573	1.7249	2.0145	0.4554
GJO	0.20379	3.4705	9.0528	0.20572	1.7290	1.7761	0.0729
WSO	0.20574	3.4705	9.0366	0.20573	1.7249	1.7753	0.0138
GPC	0.20573	3.4705	9.0366	0.20573	1.7249	1.8769	0.1656
PSOGPC	0.20573	3.4705	9.0366	0.20573	1.7249	1.8469	0.1450
GPCPSO	0.20573	3.4705	9.0366	0.20573	1.7249	1.7710	0.1022

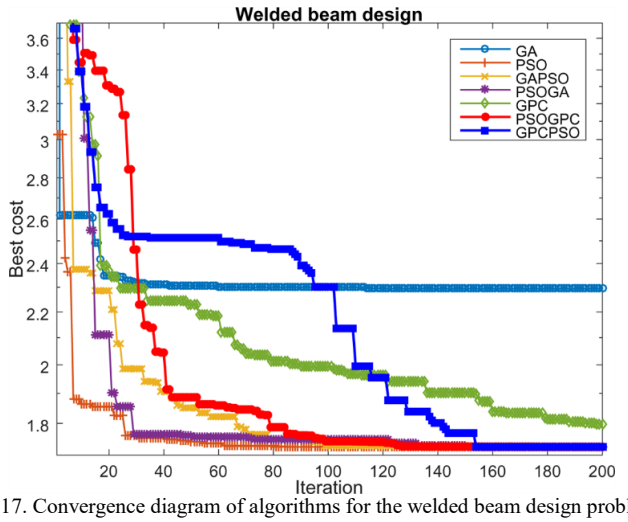

Fig 17. Convergence diagram of algorithms for the welded beam design problem

7 Conclusions

Metaheuristic hybridization algorithms combine different optimization techniques to enhance performance in solving complex problems. Research perspectives in this area focus on improving solution quality, convergence speed, and adaptability to various problem domains. Hybrid approaches often integrate local search methods with global optimization techniques, allowing for a more robust exploration of the solution space. In this paper, two competitive hybrid approaches were proposed through the combination of PSO and GPC algorithms that called competitive PSOGPC and GPCPSO. In this way, the strengths of PSO and GPC algorithms were used more effectively by using their competitive hybridization. The two proposed competitive hybrid approaches were applied and evaluated on 45 benchmark test functions. The results of experiments and statistical analysis showed that these two approaches can deal with different problems with different search spaces. The presented competitive hybrid approaches clearly showed the becoming purposeful of the search space as well as the balance between exploration and exploitation. The proposed hybrid algorithms were then applied to six classic engineering problems and compared with other hybrid methods. The results showed that these two algorithms can effectively solve engineering problems. As a discussion on research perspectives, it can be noted that the primary goal of hybridization is to leverage the strengths of multiple algorithms, leading to improved performance metrics such as solution quality and computational efficiency. Other goals are adaptability, algorithmic innovation, benchmarking, and comparisons.

One of the limitations of the research was the parameter setting. The performance of hybrid algorithms often depends heavily on the tuning of parameters, which can be a time-consuming and challenging process. In this paper, an attempt was made to consider the best settings for the proposed algorithms. Another limitation could be evaluation challenges. Assessing the performance of hybrid algorithms can be complicated due to the multitude of factors involved, including the choice of base algorithms and the specific problem context. Although this paper attempts to provide the best possible evaluation. As future work, the presented algorithms can be applied to other problems. It is also possible to make another hybridization so that the positive features of the algorithms increase even more.

References

[1] Jahani, E., & Chizari, M. (2018). Tackling global optimization problems with a novel algorithm–Mouth Brooding Fish algorithm. Applied Soft Computing, 62, 987-1002.

[2] Dehghani, M., Hubálovský, Š., & Trojovský, P. (2021). Northern goshawk optimization: a new swarm-based algorithm for solving optimization problems. Ieee Access, 9, 162059-162080.

[3] Ali, M. A., Kamel, S., Hassan, M. H., Ahmed, E. M., & Alanazi, M. (2022). Optimal power flow solution of power systems with renewable energy sources using white sharks algorithm. Sustainability, 14(10), 6049.

[4] Mei, L., & Wang, Q. (2021). Structural optimization in civil engineering: a literature review. Buildings, 11(2), 66.

[5] Gambella, C., Ghaddar, B., & Naoum-Sawaya, J. (2021). Optimization problems for machine learning: A survey. European Journal of Operational Research, 290(3), 807-828.

[6] Dahrouj, H., Alghamdi, R., Alwazani, H., Bahanshal, S., Ahmad, A. A., Faisal, A., ... & Shamma, J. S. (2021). An overview of machine learning-based techniques for solving optimization problems in communications and signal processing. IEEE Access, 9, 74908-74938.

[7] Tian, Y., Zhu, W., Zhang, X., & Jin, Y. (2023). A practical tutorial on solving optimization problems via PlatEMO. Neurocomputing, 518, 190-205.

[8] Deghbouch, H., & Debbat, F. (2021). A hybrid bees algorithm with grasshopper optimization algorithm for optimal deployment of wireless sensor networks. Inteligencia Artificial, 24(67), 18-35.

[9] Harifi, S., Khalilian, M., Mohammadzadeh, J., & Ebrahimnejad, S. (2020). New generation of metaheuristics by inspiration from ancient. In 2020 10th international conference on computer and knowledge engineering (ICCKE) (pp. 256-261). IEEE.

[10] Zhang, X., Lin, Q., Mao, W., Liu, S., Dou, Z., & Liu, G. (2021). Hybrid Particle Swarm and Grey Wolf Optimizer and its application to clustering optimization. Applied Soft Computing, 101, 107061.

[11] Dokeroglu, T., Sevinc, E., Kucukyilmaz, T., & Cosar, A. (2019). A survey on new generation metaheuristic algorithms. Computers & Industrial Engineering, 137, 106040.

[12] Rajani, B., & Kommula, B. N. (2022). An optimal energy management among the electric vehicle charging stations and electricity distribution system using GPC-RERNN approach. Energy, 245, 123180.

[13] Harifi, S. (2022). A binary ancient-inspired Giza Pyramids Construction metaheuristic algorithm for solving 0-1 knapsack problem. Soft Computing, 26(22), 12761-12778.

[14] Ebrahimnejad, S., & Harifi, S. (2022). An optimized evacuation model with compatibility constraints in the context of disability: an ancient-inspired Giza Pyramids Construction metaheuristic approach. Applied Intelligence, 52(13), 15040-15073.

[15] Harifi, S., Razavi, A., Rad, M. H., & Moradi, A. (2024). A Giza Pyramids Construction metaheuristic approach based on upper bound calculation for solving the network reliability problem. Applied Soft Computing, 167, 112241.

[16] Wu, B., Zhu, L., & Li, X. (2023). Giza pyramids construction algorithm with gradient contour approach for multilevel thresholding color image segmentation. Applied Intelligence, 1-20.

[17] Alluri, A., Lanka, R. S., & Rao, R. S. (2022). System security enhancement using hybrid HUA-GPC approach under transmission line (s) and/or generator (s) outage conditions. International Journal of Numerical Modelling: Electronic Networks, Devices and Fields, 35(3), e2970.

[18] Jaiswal, G. K., Nangia, U., & Jain, N. K. (2022). Optimal Reactive Power Dispatch Using Giza Pyramids Construction Algorithm. In 2022 Second International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT) (pp. 1-6). IEEE.

[19] Mohammadzadeh, A., Masdari, M., Gharehchopogh, F. S., & Jafarian, A. (2021). A hybrid multi-objective metaheuristic optimization algorithm for scientific workflow scheduling. Cluster Computing, 24, 1479-1503

[20] Jain, R., & Sharma, N. (2022). A quantum inspired hybrid SSA–GWO algorithm for SLA based task scheduling to improve QoS parameter in cloud computing. Cluster Computing, 1-24

[21] Tahir, A., Chen, F., Hayat, B., Shaheen, Q., Ming, Z., Ahmad, A., ... & Lim, B. H. (2023). Hybrid HP-BOA: An Optimized Framework for Reliable Storage of Cloud Data Using Hybrid Meta-Heuristic Algorithm. Applied Sciences, 13(9), 5346

[22] Dey, A., Chattopadhyay, S., Singh, P. K., Ahmadian, A., Ferrara, M., Senu, N., & Sarkar, R. (2021). MRFGRO: a hybrid meta-heuristic feature selection method for screening COVID-19 using deep features. Scientific reports, 11(1), 24065

[23] Kareem, S. S., Mostafa, R. R., Hashim, F. A., & El-Bakry, H. M. (2022). An effective feature selection model using hybrid metaheuristic algorithms for iot intrusion detection. Sensors, 22(4), 1396

[24] Al Maqbali, B. (2021). Hybrid wolf pack algorithm and particle swarm optimization algorithm for breast cancer diagnosis: Hybrid WPA and PSO algorithm. Multimedia Research, 4(3)

[25] Das, A., Guha, S., Singh, P. K., Ahmadian, A., Senu, N., & Sarkar, R. (2020). A hybrid meta-heuristic feature selection method for identification of Indian spoken languages from audio signals. IEEE Access, 8, 181432-181449

[26] Bhattacharyya, T., Chatterjee, B., Singh, P. K., Yoon, J. H., Geem, Z. W., & Sarkar, R. (2020). Mayfly in harmony: A new hybrid meta-heuristic feature selection algorithm. IEEE Access, 8, 195929-195945

[27] Yan, C., Ma, J., Luo, H., & Patel, A. (2019). Hybrid binary coral reefs optimization algorithm with simulated annealing for feature selection in high-dimensional biomedical datasets. Chemometrics and Intelligent Laboratory Systems, 184, 102-111

[28] Mafarja, M. M., & Mirjalili, S. (2017). Hybrid whale optimization algorithm with simulated annealing for feature selection. Neurocomputing, 260, 302-312

[29] Tang, C., Sun, W., Xue, M., Zhang, X., Tang, H., & Wu, W. (2022). A hybrid whale optimization algorithm with artificial bee colony. Soft Computing, 26(5), 2075-2097

[30] Zhang, X., Lin, Q., Mao, W., Liu, S., Dou, Z., & Liu, G. (2021). Hybrid Particle Swarm and Grey Wolf Optimizer and its application to clustering optimization. Applied Soft Computing, 101, 107061

[31] Sohouli, A. N., Molhem, H., & Zare-Dehnavi, N. (2022). Hybrid PSO-GA algorithm for estimation of magnetic anomaly parameters due to simple geometric structures. Pure and Applied Geophysics, 179(6-7), 2231-2254

[32] Goodarzian, F., Shishebori, D., Bahrami, F., Abraham, A., & Appolloni, A. (2023). Hybrid meta-heuristic algorithms for optimising a sustainable agricultural supply chain network considering CO2 emissions and water consumption. International Journal of Systems Science: Operations & Logistics, 10(1), 2009932

[33] Khan, A. S., Homri, L., Dantan, J. Y., & Siadat, A. (2021). Modularity-based quality assessment of a disruptive reconfigurable manufacturing system-A hybrid meta-heuristic approach. The International Journal of Advanced Manufacturing Technology, 115(5-6), 1421-1444

[34] Gupta, S., & Deep, K. (2019). Hybrid grey wolf optimizer with mutation operator. In Soft Computing for Problem Solving: SocProS 2017, Volume 2 (pp. 961-968). Springer Singapore

[35] Jahannoosh, M., Nowdeh, S. A., Naderipour, A., Kamyab, H., Davoudkhani, I. F., & Klemeš, J. J. (2021). New hybrid meta-heuristic algorithm for reliable and cost-effective designing of photovoltaic/wind/fuel cell energy system considering load interruption probability. Journal of Cleaner Production, 278, 123406

[36] Kamboj, V. K. (2016). A novel hybrid PSO–GWO approach for unit commitment problem. Neural Computing and Applications, 27, 1643-1655

[37] Teng, Z. J., Lv, J. L., & Guo, L. W. (2019). An improved hybrid grey wolf optimization algorithm. Soft computing, 23, 6617-6631

[38] Fathollahi-Fard, A. M., Dulebenets, M. A., Hajiaghaei–Keshteli, M., Tavakkoli-Moghaddam, R., Safaeian, M., & Mirzahosseinian, H. (2021). Two hybrid meta-heuristic algorithms for a dual-channel closed-loop supply chain network design problem in the tire industry under uncertainty. Advanced engineering informatics, 50, 101418

[39] Sengathir, J., Rajesh, A., Dhiman, G., Vimal, S., Yogaraja, C. A., & Viriyasitavat, W. (2022). A novel cluster head selection using Hybrid Artificial Bee Colony and Firefly Algorithm for network lifetime and stability in WSNs. Connection Science, 34(1), 387-408

[40] Al-Otaibi, S., Al-Rasheed, A., Mansour, R. F., Yang, E., Joshi, G. P., & Cho, W. (2021). Hybridization of metaheuristic algorithm for dynamic cluster-based routing protocol in wireless sensor Networksx. IEEE Access, 9, 83751-83761

[41] Agnihotri, A., & Gupta, I. K. (2018). A hybrid PSO-GA algorithm for routing in wireless sensor network. In 2018 4th International Conference on Recent Advances in Information Technology (RAIT) (pp. 1-6)

[42] Harifi, S., Mohammadzadeh, J., Khalilian, M., & Ebrahimnejad, S. (2021). Hybrid-EPC: an Emperor Penguins Colony algorithm with crossover and mutation operators and its application in community detection. Progress in Artificial Intelligence, 10(2), 181-193

[43] Kumar, M., Mukherjee, P., Verma, S., Kavita, Shafi, J., Wozniak, M., & Ijaz, M. F. (2023). A smart privacy preserving framework for industrial IoT using hybrid meta-heuristic algorithm. Scientific Reports, 13(1), 5372

[44] Akhtar, M. M., Ahamad, D., Abdalrahman, A. E. M., Shatat, A. S. A., & Shatat, A. S. A. (2022). A novel hybrid meta-heuristic concept for green communication in IoT networks: An intelligent clustering model. International Journal of Communication Systems, 35(6), e5089

[45] Seydanlou, P., Jolai, F., Tavakkoli-Moghaddam, R., & Fathollahi-Fard, A. M. (2022). A multi-objective optimization framework for a sustainable closed-loop supply chain network in the olive industry: Hybrid meta-heuristic algorithms. Expert Systems with Applications, 203, 117566

[46] Hu, H., Zhang, L., Bai, Y., Wang, P., & Tan, X. (2019). A hybrid algorithm based on squirrel search algorithm and invasive weed optimization for optimization. IEEE Access, 7, 105652-105668

[47] Abualigah, L. M., Khader, A. T., & Hanandeh, E. S. (2018). A hybrid strategy for krill herd algorithm with harmony search algorithm to improve the data clustering**1**. Intelligent Decision Technologies, 12(1), 3-14

[48] Bouaouda, A., & Sayouti, Y. (2022). Hybrid meta-heuristic algorithms for optimal sizing of hybrid renewable energy system: a review of the state-of-the-art. Archives of Computational Methods in Engineering, 29(6), 4049-4083

[49] Khalili Goudarzi, F., Maleki, H. R., & Niroomand, S. (2021). Mathematical formulation and hybrid meta-heuristic algorithms for multiproduct oil pipeline scheduling problem with tardiness penalties. Concurrency and Computation: Practice and Experience, 33(17), e6299

[50] Verma, P., & Parouha, R. P. (2021). An advanced hybrid meta-heuristic algorithm for solving small-and large-scale engineering design optimization problems. Journal of Electrical Systems and Information Technology, 8(1), 1-43

[51] Kumar, N., & Vidyarthi, D. P. (2016). A novel hybrid PSO–GA meta-heuristic for scheduling of DAG with communication on multiprocessor systems. Engineering with Computers, 32, 35-47

[52] Wang, Z., Luo, Q., & Zhou, Y. (2021). Hybrid metaheuristic algorithm using butterfly and flower pollination base on mutualism mechanism for global optimization problems. Engineering with Computers, 37, 3665-3698

[53] Aydilek, I. B. (2018). A hybrid firefly and particle swarm optimization algorithm for computationally expensive numerical problems. Applied Soft Computing, 66, 232-249

[54] Goel, R., & Maini, R. (2018). A hybrid of ant colony and firefly algorithms (HAFA) for solving vehicle routing problems. Journal of Computational Science, 25, 28-37

[55] Wahid, F., & Ghazali, R. (2019). Hybrid of firefly algorithm and pattern search for solving optimization problems. Evolutionary Intelligence, 12, 1-10

[56] Rashid, T. A., & Abdullah, S. M. (2018). A hybrid of artificial bee colony, genetic algorithm, and neural network for diabetic mellitus diagnosing. ARO-The Scientific Journal of Koya University, 6(1), 55-64

[57] Sheng, W., Shao, Q., Tong, H., & Peng, J. (2021, May). Scheduling optimization on takeout delivery based on hybrid meta-heuristic algorithm. In 2021 13th International Conference on Advanced Computational Intelligence (ICACI) (pp. 372-377). IEEE

[58] Mzili, T., Mzili, I., Riffi, M. E., & Dhiman, G. (2023). Hybrid genetic and spotted hyena optimizer for flow shop scheduling problem. Algorithms, 16(6), 265.

[59] Mzili, T., Mzili, I., Riffi, M. E., Pamucar, D., Simic, V., Abualigah, L., & Almohsen, B. (2024). Hybrid genetic and penguin search optimization algorithm (GA-PSEOA) for efficient flow shop scheduling solutions. Facta Universitatis, Series: Mechanical Engineering, 22(1), 077-100.

[60] Harifi, S., Mohammadzadeh, J., Khalilian, M., & Ebrahimnejad, S. (2021). Giza Pyramids Construction: an ancient-inspired metaheuristic algorithm for optimization. Evolutionary Intelligence, 14, 1743-1761.

[61] Surjanovic S, Bingham D (2013) Virtual Library of simulation experiments: test functions and datasets. Retrieved October 23, 2017, from http://www.sfu.ca/~ssurjano

[62] Chopra, N., & Ansari, M. M. (2022). Golden jackal optimization: A novel nature-inspired optimizer for engineering applications. Expert Systems with Applications, 198, 116924.

[63] Braik, M., Hammouri, A., Atwan, J., Al-Betar, M. A., & Awadallah, M. A. (2022). White Shark Optimizer: A novel bio-inspired meta-heuristic algorithm for global optimization problems. Knowledge-Based Systems, 243, 108457.