Evolutionary Intelligence
https://doi.org/10.1007/s12065-020-00451-3

RESEARCH PAPER

Giza Pyramids Construction: an ancient-inspired metaheuristic algorithm for optimization

Sasan Harifi[1] · Javad Mohammadzadeh[1] · Madjid Khalilian[1] · Sadoullah Ebrahimnejad[2]

Received: 4 March 2020 / Revised: 25 April 2020 / Accepted: 2 July 2020

Abstract
Nowadays, many optimization issues around us cannot be solved by precise methods or that cannot be solved in a reasonable time. One way to solve such problems is to use metaheuristic algorithms. Metaheuristic algorithms try to find the best solution out of all possible solutions in the shortest time possible. Speed in convergence, accuracy, and problem-solving ability at high dimensions are characteristics of a good metaheuristic algorithm. This paper presents a new population-based metaheuristic algorithm inspired by a new source of inspiration. This algorithm is called Giza Pyramids Construction (GPC) inspired by the ancient past has the characteristics of a good metaheuristic algorithm to deal with many issues. The ancient-inspired is to observe and reflect on the legacy of the ancient past to understand the optimal methods, technologies, and strategies of that era. The proposed algorithm is controlled by the movements of the workers and pushing the stone blocks on the ramp. This algorithm is compared with five standard and popular metaheuristic algorithms. For this purpose, thirty different and diverse benchmark test functions are utilized. The proposed algorithm is also tested on high-dimensional benchmark test functions and is used as an application in image segmentation. The results show that the proposed algorithm is better than other metaheuristic algorithms and it is successful in solving high-dimensional problems, especially image segmentation.

Keywords Metaheuristic · Optimization · Giza Pyramids Construction algorithm · GPC algorithm · Ancient-inspired · High-dimensional tests · Image segmentation · Benchmark test functions

1 Introduction

Optimization applications are numerous. Each process has the potential to be optimized. There are no companies and institutions that not involved in optimization. Many challenging applications in science and technology can be formulated as an optimization problem. Optimization can reduce time, cost and risk or increase profit, quality, and efficiency. In the industry, for example, there are cost and service quality optimization. There are also many ways to optimize time for product planning. There are many optimization issues in science, engineering, economics and business that are difficult to solve [1]. They are not solved accurately and within a reasonable time. Using the approximation algorithm is the main option to solve these problems.

Approximate algorithms are classified into two categories: heuristic and metaheuristic. Heuristics depend on the type of problem. They are usually designed and used for specific issues. Metaheuristics are more popular and are used for many issues [2]. They can be used for almost any kind of problem. The metaheuristic solves problems that seem to be difficult, by searching for a large space of solutions. These algorithms achieve this goal by effectively exploring space and reducing the size of the solution space. Metaheuristics solve problems faster, solve bigger problems, and are stronger algorithms. They are also very flexible and easy to design and implement.

The metaheuristic is a branch of optimization in computer science and applied mathematics that deals with complex

✉ Sasan Harifi
s.harifi@kiau.ac.ir

Javad Mohammadzadeh
j.mohammadzadeh@kiau.ac.ir

Madjid Khalilian
khalilian@kiau.ac.ir

Sadoullah Ebrahimnejad
ibrahimnejad@kiau.ac.ir

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

[2] Department of Industrial Engineering, Karaj Branch, Islamic Azad University, Karaj, Iran

Published online: 13 July 2020

computation theory and algorithms. It also covers other areas such as artificial intelligence, computational intelligence, soft computing, mathematical programming, and operations research. In practice, metaheuristics have been very effective in solving complex real-world problems, and have also played a significant role in reducing costs and expanding rapidly in various fields [3].

The metaheuristics include the following categories. These are Evolutionary-based, Trajectory-based, and Nature-inspired methods. We do not claim that this categorization is unique. Some algorithms can fall into several different categories.

Evolutionary models are an abstract model of biological evolution in which populations that are considered candidates for a solution are frequently exposed to natural selection or genetic diversity [4]. Evolutionary is modeled on the concept of competition. This type of metaheuristics simulates species evolution. They are based on the evolution of the population. Populations are usually randomly generated, and each individual is actually a solution. An objective function determines which solution is appropriate. At each stage according to the pattern of choice, anyone who has better suitability is selected with a higher probability. Then the population selected are reproduced by different operators to create a new generation of offspring. These operators are usually crossover and mutation. Finally, an alternative plan is applied to determine which of the offspring and parents will remain. Some of the popular evolutionary-based techniques are Genetic Algorithm (GA) [5], Memetic Algorithm (MA) [6], Differential Evolution (DE) [7], Harmony Search (HS) [8], Clonal Selection Algorithm (CSA) [9], and so on.

Trajectory-based are methods that work on a solution. In this way, the problem space is searched with a path [10]. The characteristics of these methods are usually determined by the type of problem. Trajectory-based metaheuristics improve a single solution. These are carried out with repetitive routines through which they move from one solution to another. It has shown its effectiveness in a variety of issues. Some of the popular techniques in this category are Simulated Annealing (SA) [11], Tabu Search (TS) [10], Iterated Local Search (ILS) [12], Guided Local Search (GLS) [13], Greedy Randomized Adaptive Search Procedure (GRASP) [14], Variable Neighborhood Search (VNS) [15], and so on.

Nature-inspired are methods that follow the laws of nature. Nature has simple and understandable laws. In fact, the collective behavior of most organisms is as a seeker in the problem space that leads to the goal and solution [16]. Nature-inspired itself contains Swarm-based, Bio-inspired, Physics/Chemistry-based, Human-based, and Plant-based methods. Swarm-based is about group behaviors and swarm intelligence of a set of beings. Swarm can be defined as an organized set of agents or entities that work together. It can be an example of the lives of ants, bees, termites, fish, and birds. Many swarm-based algorithms are presented such as Particle Swarm Optimization (PSO) [17], Firefly Algorithm (FA) [18], Artificial Bee Colony (ABC) [19], Ant Colony Optimization (ACO) [20], Emperor Penguins Colony (EPC) [21], and so on. It is true that swarm intelligence algorithms can be considered as a subset of bio-inspired algorithms. But many bio-inspired algorithms do not directly use swarm behavior. So it is better to put them in a separate subset namely Bio-inspired. The Krill Herd Algorithm (KHA) [22], Crow Search Algorithm (CSA) [23], Grey Wolf Optimizer (GWO) [24], Owl Search Algorithm (OSA) [25], Ant Lion Optimizer (ALO) [26], and so on are examples of bio-inspired algorithms. As mentioned earlier, sometimes algorithms can be in several categories. Physics/Chemistry-based are mostly created by imitating the laws of physics or chemistry. These features and laws include electrical charge, gravity, physical and chemical changes of materials, and so on. From this subset we can refer to Chemical Reaction Optimization (CRO) [27], Black Hole (BH) [28], Multi-Verse Optimizer (MVO) [29], Thermal Exchange Optimization (TEO) [30], and so on. Human-based methods models all the individual and social behaviors of human. This subset includes individual behaviors such as how humans search for and adapt to the environment, and how human beings behave with different emotions. Social behaviors, such as the chaos that occurs in society, the way human beings work together or even imperialist can also be modeled. The most popular algorithms in this subset are Imperialist Competitive Algorithm (ICA) [31], Cultural Algorithm (CA) [32], Teaching–Learning-Based Optimization (TLBO) [33], and recently published Political Optimizer (PO) [34]. Plant-based methods are inspired by the process of plant growth, plant dispersal, root and plant expanding, and so on. In general, any algorithm that somehow models a plant falls into this subset. Example of algorithm in this subset are Invasive Weed Optimization (IWO) [35], and Artificial Root Foraging Algorithm (ARFA) [36].

The ancient past is the new ideology and origin of inspiration that introduced by this paper for the first time. In the ancient past there were numerous limitations, but various man-made structures show that the limitations and the lack of hardware and software facilities have given rise to some sort of optimization in ancient times. The Giza Pyramids Construction[1] (GPC) optimization algorithm is also presented as the first ancient-inspired metaheuristic algorithm. Figure 1 shows the classification of metaheuristics explained in this paper. In this figure, the existing classifications and the position of the new ancient-inspired

[1] Source codes of GPC are publicly available at www.harifi.com.

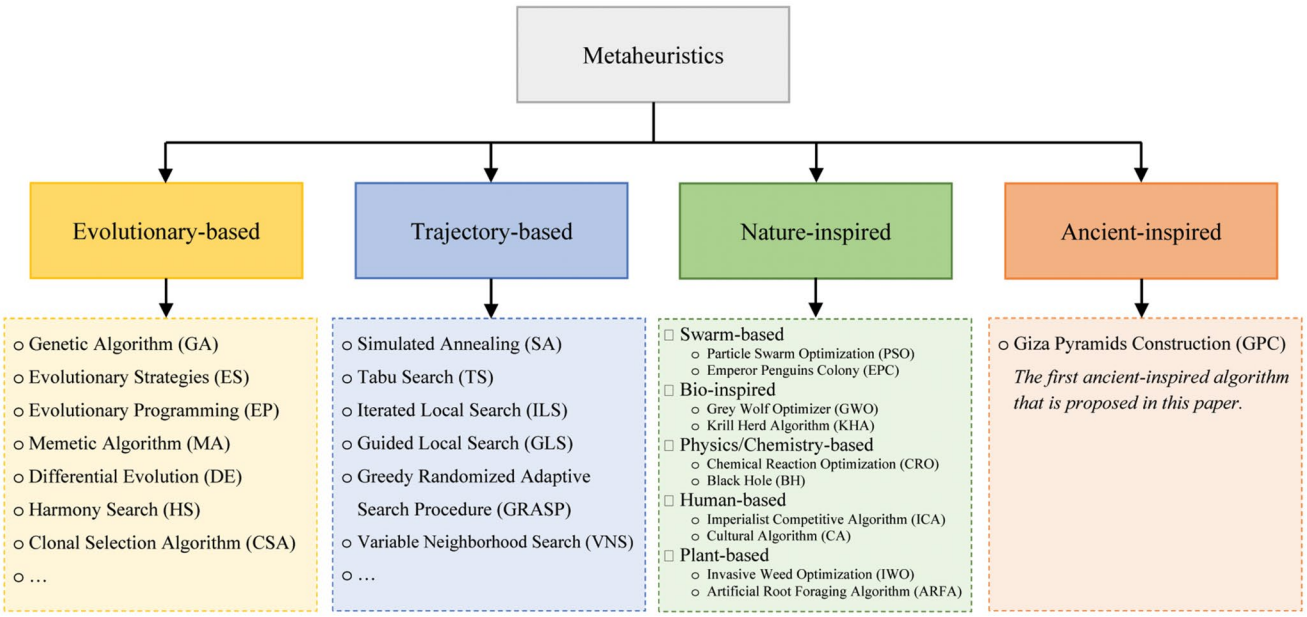

Fig. 1 Classification of metaheuristic algorithms

category are shown. This new inspiration can include all the features of the existing inspirations.

The main contribution and innovation of this paper is the use of a novel and efficient source of inspiration compared to existing sources of inspiration. This novel source of inspiration is actually a combination of evolution and nature with many new features that the existence of limitation in the ancient past has led to the flourishing of its capabilities. The main goal of this study is to provide a new metaheuristic algorithm based on this novel source of inspiration, as well as to test and apply it to a specific application to show that this approach is feasible and reliable.

As mentioned earlier, metaheuristics are used in almost every field. These fields range from computer science [2] to industry [37] or civil engineering [38]. Their applications are also varied, for example in computer science they can be used to optimize fuzzy systems [39], feature selection [40, 41], information retrieval [42], and so on. In this paper as an application, we decided to apply the proposed algorithm in image segmentation. Image segmentation is the process of dividing an image into homogeneous areas can be considered a high-dimensional problem. Segmentation is important because it prepares the image for applications such as machine vision, and image analysis including medical images, satellite images, and so on.

Therefore, the experiments performed in this paper are divided into two parts. The first is related to the standard experiment using benchmark test functions, which are performed in two ways: low-dimensions and high-dimensions. The second is related to the experiment of application in image segmentation, which is itself an example of a high-dimensional problem.

The remainder of this paper is structured as follows: Sect. 2 introduces ancient-inspired ideology. Section 3 describes Giza Pyramids Construction (GPC) algorithm. Section 4 includes experimental results and discussion. Section 5 provides the statistical analysis. Section 6 represents high-dimensional tests. Section 7 presents application in image segmentation. Finally, Sect. 8 represents conclusions.

2 Ancient-inspired ideology

Ancient refers to past events from the beginning of the writing and recording of the history of humanity to the beginning of the post-classical period [43, 44]. The length of this period is approximately 5000 years. Historians consider the end of this period to be 500 AD. There are two ways to better understand this era. The first is through archeology. Archeology is the exploration and study of ancient artifacts to interpret and reconstruct human past behavior [45]. Archaeologists are searching the ruins of ancient cities to find clues about the lifestyle and the time period [46]. The second way is through studying textual sources. Textual sources are narratives of ancient historians [47]. Many of the events described by them are based on an understanding of ancient history.

In the ancient past, productive work enabled humans to gain wealth. In fact, work was a religious duty and a moral virtue in public morality. The technical skills were highly commended. Work and technical skills were the core of human intelligence

and the key to understanding nature. This core and this key gradually led to the emergence of superior technologies in the ancient past [48]. One of the most attractive parts of ancient times is the study of ancient technology and science. In history, technology and ancient science evolved during the development of ancient civilizations and technological advances in engineering [49]. Among the technologies can be mentioned such as Egyptians, Indians, Chinese, Greeks, Romans and Iranians technologies.

The fact is that the advanced civilizations mentioned created the knowledge infrastructures on a large scale so that these infrastructures were not found elsewhere. This knowledge infrastructure has evolved into an integrated network with the spread of civilizations, where science and technology have been easily circulated. In these civilizations, the elites have always strived to acquire each other's technical knowledge [50]. An essential element in the development of antiquity construction has been the use of pre-manufactured standard-size materials. This greatly reduced construction costs and improved development time. In ancient Egyptian civilization, for example, stones were brought to the site to prepare the pyramids [50]. Technology in the ancient past was always used to optimize and reduce costs. Certainly, inventions such as concrete and standard building materials would definitely cost less for construction projects and give more freedom to the architects of the structural design. Also, innovations in construction could create large-scale complex structures with low cost. Standardization of building materials and the creation of manufacturing technologies could even reduce labor costs [50].

In general, the emergence of construction techniques and the use of new strategies by builders has been a major innovation that has had a profound impact on the social, economic and cultural history of ancient civilizations. The question, however, is how the workforce management strategy has been and how it has been used. Looking at some of the monuments and artifacts from antiquity and estimating the time of construction of these monuments and considering the emergence of new technology and the lack of advanced technology at that time, it raises the question that how was the optimization strategies in construction these buildings and monuments [50]. The ancient-inspired ideology is the observation and thinking about the remnants and legacies of antiquity and the understanding of the management and strategy used in that era to advance civilization and optimize and reduce the cost of living.

3 Giza Pyramids Construction (GPC) algorithm

The Giza Pyramids complex, also known as Giza Necropolis, is a site that has three large pyramids, all built during the fourth dynasty of ancient Egypt [51]. The largest pyramid, also known as the Seven Wonders, is called the Khufu Pyramid. The other two pyramids are named Khafre and Menkaure [52]. According to archaeologists, the method of construction is different from each other because construction of pyramids have been developed over time. How to manage the workers was the most important issue in building the pyramids. Due to the lack of hardware facilities, relatively short construction time and the large number of stone blocks used in the pyramids, its construction has been optimized. In this section, the construction method, the inspiration, and the proposed algorithm are described in detail.

3.1 The construction

Many theories have been put forward about the pyramids construction methods, none of which is hundred percent approved. Many believe that the stones of this pyramids were removed from mines, shipped and then placed in place. Such a way that, they used ramps to mount them at higher levels [53, 54]. The Greeks believed that slaves were exploited in the construction of the pyramids, but new findings suggest that the creators of the pyramids were skilled workers who lived well in the surrounding villages. The labor force needed to build the pyramids was an average of fourteen thousand people and a maximum of forty thousand [55]. Without the use of metal tools, wheels and other tools, these pyramids were built over a period of 10–20 years. The number of blocks used in the largest pyramid is two million pieces. The workers, including slaves, coolies, masons, metalworkers, carpenters, and foreman.

In addition to the challenges of project construction, there is a need for an advanced approach to project management, construction management and method of construction. The project involved building a prominent construction plant. Issues such as feeding, housing, the payment of workers' salaries, and work scheduling for timely completion before the death of the pharaoh were also considered [56]. The pyramid stands today as awesome testimony to the skill and sheer determination of the ancient race that built it. The complexity and logistical requirements of this project are simply extraordinary.

3.2 The inspiration

As noted, the workers are slaves, coolies, masons, metalworkers, carpenters, led by an expert agent. This expert agent is a foreman called Pharaoh's special agent. Workers carry stone blocks. This is done under the direct supervision of the Pharaoh's agent. There may be different workers each responsible for carrying a stone block. The task report should be regularly given to Pharaoh's special agent. This

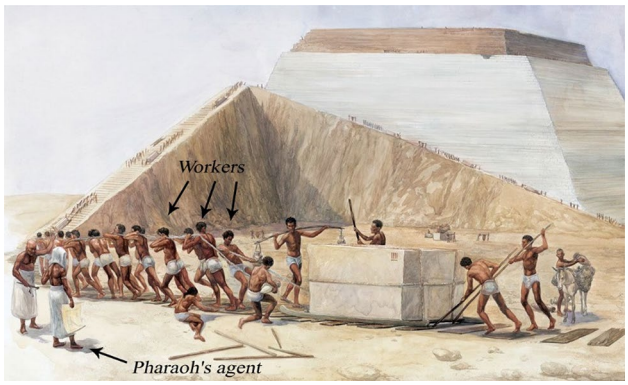

Fig. 2 Subjective perception of Pharaoh's special agent and workers

task is performed by each worker. Figure 2 illustrates an idea of how these factors do the job, namely the simple worker, and the Pharaoh's special agent.

The workers at the construction site each have a position. A worker who does his job better will receive sublime rank as a reward. So there is a competition to get sublime rank. The best rank is related to Pharaoh's special agent. Also, the energy lost during the stone block transport phase may cause workers to rest for some time. If a worker loses too much power or gets tired, he will be substituted by young blood and energetic workers. This means that if there is no improvement in the workforce, it must be substituted with a new one. The weak worker may be more efficient elsewhere. In addition to competing for sublime rank, there is another motivational competition among workers as each can gain experience and expertise.

During the pyramid construction process, stone blocks that are scattered by agents such as miners around the construction site are carried by the workers to the construction site. This is done every day so that the stone blocks are collected from around and dragged to the pyramid. Ramps were used to build the pyramid. The distance between the stone block and the location of its installation in the pyramid must be traveled. The distance traveled to push the stone is measured by the ability of the workers. During the workday, if enough power is available, more shipment will be carried out by the workers, making the block closer to the installation site in the pyramid. It should be noted, the ramp gradient, initial velocity, and friction force influence the movement of the stone block.

3.3 The proposed algorithm

Suppose that stone blocks are scattered around the construction site. Workers have to push the stone blocks to the installation place. The initial position of each block and its cost are known. Ramps are used to move the stone

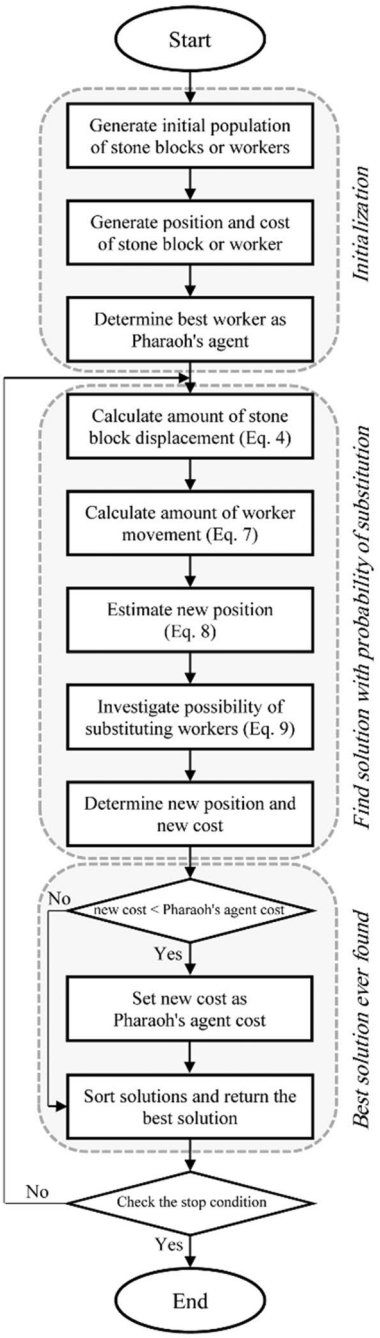

Fig. 3 Flowchart of the proposed GPC

blocks to the installation place. The slope of the ramp and its friction affect the displacement of the stone blocks. Also, the workers themselves are constantly moving to find the best position to dominate the stone block. Due to the different characteristics of each worker, it is possible to substitute workers to balance the power of the workers for carrying a stone block. Therefore, the position of some workers will be substituted with others. This substituting

causes a change in the stone block displacement system and the power balance. Algorithm 1 describes pseudo-code of the GPC algorithm. Also, Fig. 3 shows the flow-chart of the algorithm. For this algorithm, there are some rules as follows:

1. The pyramids were built by using a straight-on ramp.
2. It is assumed that only one ramp is used.
3. In the algorithm, the angle the ramp makes with the horizon is less than 15° and can be variable. (Archaeologists believe that the angle was between 8° and 12° [57]).
4. The solutions are derived from the resultant of worker position and stone block. Because the worker is actually pushing the stone block.
5. Friction is effective in the displacement of a stone block but is not considered for workers.
6. During the construction process, some workers are probably to be substituted and put into a new position.

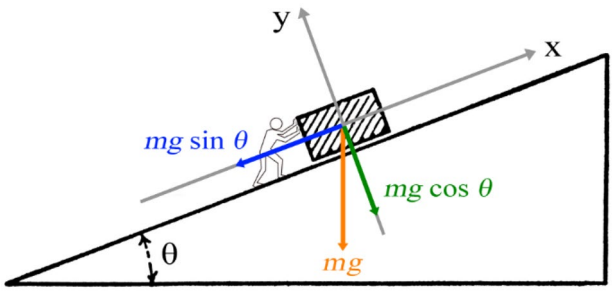

Fig. 4 Position of the object and the coordinate axis on the ramp

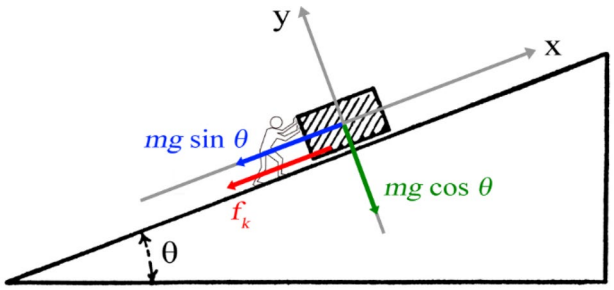

Fig. 5 The forces acting on the object

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

```
STEP 1:
   generate initial population array of stone blocks or workers (Population size);
   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. 4);
         calculate amount of worker movement (Eq. 7);
         estimate new position (Eq. 8);
         investigate possibility of substituting workers (Eq. 9);
         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
```

Here we need the equations for movement of an object on the inclined plane or ramp. Given that the object (stone block) moves along the inclined plane or ramp, the coordinate axis is adjusted so that the direction of the x-axis is in the direction of acceleration. By selecting this coordinate axis, the force applied to the object is not positioned horizontally and vertically. In this way, we have the mass force on the axis. The illustration is shown in Fig. 4. And from now on, instead of the mass, we plot the forces that have been decomposed from the mass.

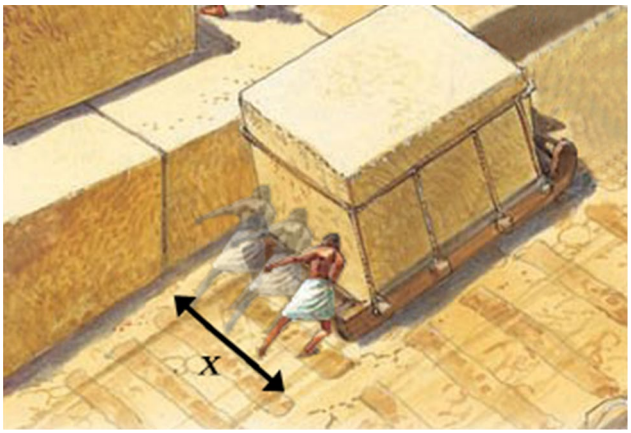

Fig. 6 Subjective perception of worker moving or shaking to gain the best dominate and best control of the stone block

Now, given the essence of the algorithm, each stone block is pushed upward by the initial velocity v_0 on the ramp. Therefore, the stone block stops after traversing distance d on the ramp. The forces applied to the stone block are illustrated in Fig. 5.

As shown in Fig. 5, f_k is the kinetic friction force, and since the stone block is at the threshold of displacement, f_k can be obtained from the following equation by the general equation of maximum static friction force,

$$f_k = \mu_k mg\cos\theta \tag{1}$$

where m is the mass of the stone block, g is the gravity of the earth, θ is the angle that the ramp makes with the horizon, and μ_k is the kinetic friction coefficient. Because we are on the x-axis, according to Newton's second law namely $\sum \vec{F} = m\vec{a}$, we have,

$$-mg\sin\theta - f_k = ma \tag{2}$$

where a is acceleration. By placing the Eq. 1 in the Eq. 2, the acceleration of the stone block upward on the ramp is obtained. Therefore, we have,

$$a = -g\left(\sin\theta + \mu_k\cos\theta\right) \tag{3}$$

Here, we need a time-independent equation of motion under constant acceleration, which can be obtained using the following equation to calculate the displacement of a stone block on the ramp,

$$d = \frac{v_0^2}{2g\left(\sin\theta + \mu_k\cos\theta\right)} \tag{4}$$

where d is the value of displacement. g is gravity the Earth as mentioned earlier. The value of g is 9.8. θ is the angle that the ramp makes with the horizon. v_0 is the initial velocity of the stone block and in the algorithm is determined by a uniformly distributed random number in each iteration. With this arrangement, if a worker applies force to a stone block, the stone block starts moving at an initial velocity. According to the physical sciences, the force of friction causes the stone block to stop after a while. So the worker applies another force on the stone block again so that the stone block starts moving again at an initial velocity. In each iteration of the algorithm, the initial velocity is considered a random number, because each time the worker attempts to move the stone block, the applied force varies according to the power consumed by the worker. Thus, for v_0 we have,

$$v_0 = rand(0,1) \tag{5}$$

In fact, $rand(0,1)$ is a random number between 0 and 1, namely $0 < v_0 = rand(0,1) < 1$. Also, μ_k is the kinetic coefficient of friction between the stone block and the ramp and in the algorithm is determined by the uniformly distributed random number. So for μ_k, we have,

$$\mu_k = rand\left[\mu_{k\_min}, \mu_{k\_max}\right] \tag{6}$$

In the algorithm, the minimum μ_k and the maximum μ_k are predetermined, then a random number between these two values is assumed in each iteration. In the other words, μ_k is $\mu_{k\_min} \le \mu_k \le \mu_{k\_max}$. The reason for the randomness of the amount of friction is that the ramp surface is not polished, and due to possible unevenness in some parts, friction may increase or decrease.

The basic idea of the algorithm is that the workers pushing the stone block are constantly moving or shaking to gain the best dominate and best control of the stone block. These shocks cause the worker to perform non-repetitive movements to push the stone block better. Figure 6 illustrates the subjective perception of this movement.

As mentioned earlier, the Eq. 4 determines the amount of stone block displacement relative to its previous position. This equation is used with little change to determine the new position of the worker. For the worker, friction is not considered. Thus, the new position of the worker pushing the stone block is obtained from the following equation,

$$x = \frac{v_0^2}{2g\sin\theta} \tag{7}$$

that is the Eq. 4 regardless of friction. In this equation, x is the amount of worker movement as shown in Fig. 6. The worker moves upwards with the stone block and simultaneously applies force to the stone block. The goal here is for workers to have better control over the rock block with their small motions. But the worker also has the initial velocity. So for the worker, the friction is not considered. As mentioned

earlier, this is one of the rules for the algorithm. After calculating the changes of stone block displacement and worker movement through the Eqs. 4 and 7, a new position can be obtained from the resultant of these two equations. This new position is a new solution. So, in the algorithm to get a new solution, we have,

$$\vec{p} = \left(\vec{p}_i + d\right) \times x\vec{\epsilon}_i \tag{8}$$

where in the Eq. 8, $\vec{p_i}$ is the current position, d is the displacement value of the stone block (Eq. 4), x is the amount of worker movement (Eq. 7), and $\vec{\epsilon_i}$ is a random vector that follows the Uniform, Normal or Lévy distribution. In this way, the new position is obtained by adding the displacement of the stone block to the previous position, which is multiplied by the amount of worker displacement. Multiplying this value determines the position of the worker around the stone block for the next iteration.

Sometimes, during the construction of the pyramids, the worker lost his ability or lose his power, as a result, he was substituted by another. This substituting was done to balance the power. In this way, the workers were used to carry the fitted stone block in their own accord, given their strength and abilities. So a worker who has not efficiency in his own position, maybe in some other positions, efficiency will come. This substituting operation is performed in the algorithm with fifty percent probability (by default). Therefore, there is a fifty percent chance of one worker being substituted with another in each iteration. The use of substituting operation is very similar to a uniform crossover operator. It is assumed, if the primary solutions of the problem are $\boldsymbol{\Phi} = \left(\varphi_1, \varphi_2, \ldots, \varphi_n\right)$ and the generated solutions using the Eq. 8 are $\boldsymbol{\Psi} = \left(\psi_1, \psi_2, \ldots, \psi_n\right)$, with a fifty percent probability, some of the primary solutions will be substituted with the generated solutions. So we will have new solutions, $\mathrm{Z} = \left(\zeta_1, \zeta_2, \ldots, \zeta_n\right)$. As such, the following relationship is used to substitution,

$$\zeta_k = \begin{cases} \psi_k, \ \textit{if } \textit{rand}[0,1] \leq 0.5 \\ \varphi_k, \ \textit{otherwise} \end{cases} \tag{9}$$

4 Experimental results and discussion

This section describes the experiments performed to evaluate the performance of the GPC algorithm. In the experiments, thirty standard benchmark test functions [58] were used to evaluate performance. In applied mathematics, the test functions are known as artificial landscapes. These functions are useful for evaluating the characteristics of optimization algorithms. Also, these functions can be used to evaluate the accuracy, performance, efficiency and convergence rate of optimization algorithms. The benchmark functions used are presented in Table 1.

For validation and performance evaluation, the proposed algorithm is compared with five improved and popular algorithms. These algorithms are Genetic Algorithm (GA), Imperialist Competitive Algorithm (ICA), Particle Swarm Optimization (PSO), Differential Evolution (DE), and Firefly Algorithm (FA).

In order that, the experiment results are comparable, the settings of all algorithms are similar to each other. For this purpose, the initial population for all algorithms is considered 20. Also, the number of decision variables for all algorithms is considered 30. Decision variables also determine the dimensions of the problem. The experiment process is performed as each algorithm is run 50 times and the mean and standard deviation of these 50 independent runs are recorded. Some parameters of some algorithms are tune-up manually. For example, the crossover and mutation rates in the GA are tune-up manually to obtain the best solutions. It should be noted that the type of crossover used in the implemented DE algorithm is binomial crossover. Also, the crossover and mutation type used in the implemented GA are arithmetic crossover and Gaussian mutation, respectively. The values of all parameters for each algorithm are shown in Table 2. The stop condition of all algorithms is the Number of Function Evaluations (NFE). This means that in our experiments, the algorithms stop after NFE calls and the results are recorded. This allows the algorithms to be compared in equal terms if the algorithms differ in time complexity. All evaluation experiments have been run on an Intel® Pentium® processor CPU G645 2.90 GHz with 2 GB RAM. Implementations have been run on MATLAB R2015b for coding.

The performance results of the algorithms are shown in Table 3. Here we have to explain that for some functions the number of calls is 1000 (10^3) NFE calls and for others, it is 10,000 (10^4) NFE calls. The difference is because some test functions are difficult and require more NFE calls to get better solutions. The results show that in general, the GPC algorithm is more successful than other algorithms in finding optimal solutions.

The Bohachevsky $\left(f_3\right)$ and Booth $\left(f_4\right)$ functions are convex, unimodal, and defined for two-dimensional space. The best solutions for these functions are related to the GPC algorithm and after the GPC algorithm, the DE algorithm provides the best solutions. For the Bukin (f_5) function, which is multimodal, non-differentiable, non-separable and defined for two-dimensional space, the GPC algorithm provides the best solutions. Matyas $\left(f_{19}\right)$, Sphere $\left(f_{27}\right)$, and Sum-Squares $\left(f_{29}\right)$ functions are almost simple functions, these functions are convex, unimodal and differentiable, except that Matyas is non-separable while Sphere and Sum-Squares are separable. For these three functions, the best

Table 1 Standard benchmark test functions

Function name	Equation	Range	$f(x*)$
Ackley	$f_1(x) = -20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos\left(2\pi x_i\right)\right) + 20 + \exp$	$[-32.768, 32.768]$	0
Beale	$f_2(x) = \left(1.5 - x_1 + x_1x_2\right)^2 + \left(2.25 - x_1 + x_1x_2^2\right)^2 + \left(2.625 - x_1 + x_1x_2^3\right)^2$	$[-4.5, 4.5]$	0
Bohachevsky	$f_3(x) = x_1^2 + 2x_2^2 - 0.3\cos\left(3\pi x_1\right) - 0.4\cos\left(4\pi x_2\right) + 0.7$	$[-100, 100]$	0
Booth	$f_4(x) = \left(x_1 + 2x_2 + 7\right)^2 + \left(2x_1 + x_2 + 5\right)^2$	$[-10, 10]$	0
Bukin	$f_5(x) = 100\sqrt{\left\lvert x_2 - 0.01x_1^2\right\rvert} + 0.01\left\lvert x_1 + 10\right\rvert$	$[-3, 3]$	0
Camel Six-Hump	$f_6(x) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + \left(-4 + 4x_2^2\right)x_2^2$	$[-3, 3]$	-1.0316
Camel Three-Hump	$f_7(x) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1x_2 + x_2^2$	$[-5, 5]$	0
Cross-In-Tray	$f_8(x) = -0.0001\left(\left\lvert\sin\left(x_1\right)\sin\left(x_2\right)\exp\left(\left\lvert 100 - \frac{\sqrt{x_1^2+x_2^2}}{\pi}\right\rvert\right)\right\rvert + 1\right)^{0.1}$	$[-10, 10]$	-2.06261
De Jong	$f_9(x) = \left(0.002 + \sum_{i=1}^{25}\frac{1}{i+\left(x_1-a_{1i}\right)^6+\left(x_2-a_{2i}\right)^6}\right)^{-1}$	$[-65.536, 65.536]$	0
Dixon Price	$f_{10}(x) = \left(x_1 - 1\right)^2 + \sum_{i=2}^{d} i\left(2x_i^2 - x_{i-1}\right)^2$	$[-10, 10]$	0
Drop-Wave	$f_{11}(x) = -\frac{1+\cos\left(12\sqrt{x_1^2+x_2^2}\right)}{0.5\left(x_1^2+x_2^2\right)+2}$	$[-5.12, 5.12]$	-1
Easom	$f_{12}(x) = -\cos\left(x_1\right)\cos x_2\exp(-\left(x_1 - \pi\right)^2 - (x_2 - \pi)^2)$	$[-100, 100]$	-1
Eggholder	$f_{13}(x) = -\left(x_2 + 47\right)\sin\left(\sqrt{\left\lvert x_2 + \frac{x_1}{2} + 47\right\rvert}\right) - x_1\sin\left(\sqrt{\left\lvert x_1 - \left(x_2 + 47\right)\right\rvert}\right)$	$[-512, 512]$	-959.6407
Griewank	$f_{14}(x) = \sum_{i=1}^{d}\frac{x_i^2}{4000} - \prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]$	0
Holder-Table	$f_{15}(x) = -\left\lvert\sin\left(x_1\right)\cos(x_2)\exp\left(\left\lvert 1 - \frac{\sqrt{x_1^2+x_2^2}}{\pi}\right\rvert\right)\right\rvert$	$[-10, 10]$	-19.2085
Hyper-Ellipsoid	$f_{16}(x) = \sum_{i=1}^{d}\sum_{j=1}^{i} x_j^2$	$[-65.536, 65.536]$	0
Levy	$f_{17}(x) = \sin^2\left(\pi\omega_1\right) + \sum_{i=1}^{d-1}\left(\omega_i - 1\right)^2\left[1 + 10\sin^2\left(\pi\omega_i + 1\right)\right] + \left(\omega_d - 1\right)^2\left[1 + \sin^2\left(2\pi\omega_d\right)\right]$	$[-10, 10]$	0
Levy N.13	$f_{18}(x) = \sin^2\left(3\pi x_1\right) + \left(x_1 - 1\right)^2\left[1 + \sin^2\left(3\pi x_2\right)\right] + \left(x_2 - 1\right)^2\left[1 + \sin^2\left(2\pi x_2\right)\right]$	$[-10, 10]$	0
Matyas	$f_{19}(x) = 0.26\left(x_1^2 + x_2^2\right) - 0.48x_1x_2$	$[-10, 10]$	0
Michalewicz	$f_{20}(x) = -\sum_{i=1}^{d}\sin(x_i)\sin^{2m}\left(\frac{ix_i^2}{\pi}\right)$	$[0, \pi]$	-1.8013
Powell	$f_{21}(x) = \sum_{i=1}^{d/4}\left[\left(x_{4i-3} + 10x_{4i-2}\right)^2 + 5\left(x_{4i-1} + x_{4i}\right)^2 + \left(x_{4i-2} + 2x_{4i-1}\right)^4 + 10\left(x_{4i-3} + x_{4i}\right)^4\right]$	$[-4, 5]$	0
Rastrigin	$f_{22}(x) = 10d + \sum_{i=1}^{d}\left[x_i^2 - 10\cos\left(2\pi x_i\right)\right]$	$[-5.12, 5.12]$	0
Rosenbrock	$f_{23}(x) = \sum_{i=1}^{d-1}\left[100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2\right]$	$[-5, 10]$	0
Schaffer N.2	$f_{24}(x) = 0.5 + \frac{\sin^2\left(x_1^2-x_2^2\right)-0.5}{\left[1+0.001\left(x_1^2+x_2^2\right)\right]^2}$	$[-100, 100]$	0
Schaffer N.4	$f_{25}(x) = 0.5 + \frac{\cos\left(\sin\left(\left\lvert x_1^2-x_2^2\right\rvert\right)\right)-0.5}{\left[1+0.001\left(x_1^2+x_2^2\right)\right]^2}$	$[-100, 100]$	0.292579
Shubert	$f_{26}(x) = \left(\sum_{i=1}^{5} i\cos((i+1)x_1 + i)\right)\left(\sum_{i=1}^{5} i\cos((i+1)x_2 + i\right))$	$[-10, 10]$	-186.7309
Sphere	$f_{27}(x) = \sum_{i=1}^{d} x_i^2$	$[-5.12, 5.12]$	0
Sum-Powers	$f_{28}(x) = \sum_{i=1}^{d}\left\lvert x_i\right\rvert^{i+1}$	$[-1, 1]$	0
Sum-Squares	$f_{29}(x) = \sum_{i=1}^{d} ix_i^2$	$[-10, 10]$	0
Zakharov	$f_{30}(x) = \sum_{i=1}^{d} x_i^2 + \left(\sum_{i=1}^{d} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{d} 0.5ix_i\right)^4$	$[-5, 10]$	0

Table 2 The values used to adjust the parameters of the algorithms

Algorithm	Parameters	Values
GA	Population size	20
	Crossover percentage	0.8
	Mutation percentage	0.3
	Mutation rate	0.02
	Selection pressure	8
ICA	Population size	20
	Number of empires/imperialists	10
	Selection pressure	1
	Assimilation coefficient	1.5
	Revolution probability	0.05
	Revolution rate	0.1
	Colonies mean cost coefficient	0.2
PSO	Swarm size	20
	Inertia weight	1
	Inertia weight damping ratio	0.99
	Personal learning coefficient	2
	Global learning coefficient	2
DE	Population size	20
	Lower bound of scaling factor	0.2
	Upper bound of scaling factor	0.8
	Crossover probability	0.2
FA	Swarm size	20
	Light absorption coefficient	1
	Attraction coefficient base value	2
	Mutation coefficient	0.2
	Mutation coefficient damping ratio	0.98
GPC	Population size	20
	Gravity	9.8
	Angle of ramp	10
	Initial velocity	rand(0, 1)
	Minimum friction	1
	Maximum friction	10
	Substitution probability	0.5

solutions are related to the GPC algorithm. The solutions provided by GPC are very different from other algorithms. Also in the case of the Sum-Powers (f_{28}) function, which is a convex, unimodal, non-differentiable and separable function, the best solution with a lot of difference is related to the GPC algorithm.

In the case of the two functions Hyper-Ellipsoid (f_{16}) and Zakharov (f_{30}), which are convex, unimodal and defined for n-dimensional space, the best solutions are also related to the proposed GPC algorithm. The three functions Dixon Price (f_{10}), Rastrigin (f_{22}), and Rosenbrock (f_{23}) are difficult, convex, multimodal, differentiable, and defined for the n-dimensional space. For these three functions, GPC also provides the best solutions. In the case of the Rastrigin function, the GPC algorithm finds the best possible optimal solution in every 50 independent runs. The functions of Griewank (f_{14}), Schaffer N.2 (f_{24}), and Schaffer N.4 (f_{25}) are non-convex, unimodal, differentiable, non-separable, and defined for two-dimensional space. The GPC provides the best solutions for these functions with the explanation that for the Schaffer N.2 function the best possible optimal solution is found for all 50 independent runs.

The functions Ackley (f_1), Beale (f_2), Camel Six-Hump (f_6), Camel Three-Hump (f_7), Easom (f_{12}), Levy (f_{17}), Levy N.13 (f_{18}), and Shubert (f_{26}), all are non-convex, multimodal, differentiable, and non-separable functions. These functions are defined for two-dimensional space and some for n-dimensional space. These functions fall into the category of complex functions. In the case of the Camel Six-Hump function, the solutions of ICA, DE and GPC algorithms are jointly better. In the case of the Levy N.13 function, the solutions obtained from the DE algorithm are better than the other algorithms. Cross-In-Tray (f_8) and Holder-Table (f_{15}) functions are non-convex, multimodal, non-differentiable and non-separable. For Holder-Table, the best solution is provided by GPC, but for Cross-In-Tray function, the ICA, DE, and GPC are jointly better.

Other functions such as De Jong (f_9), Drop-Wave (f_{11}), Michalewicz (f_{20}), and Powell (f_{21}) are multimodal and non-convex functions. For these functions, the best answers with the most differences are related to the proposed GPC algorithm. Finally, for the Eggholder (f_{13}) function, which is a difficult function to optimize because it has a large local optimal value, the best solutions are provided by the GPC algorithm. The FA algorithm has the best solution for this function after GPC. Overall, out of the 30 test functions, the proposed GPC algorithm provides the best solutions for the 27 functions. Also, it provides a jointly better solution for the two functions, and it does not provide the best solution for only one function in comparison with other algorithms.

If we want to calculate the computational complexity of the algorithm, both the time and space complexity must be considered. In the GPC algorithm, the main loop, which counts the number of iterations, includes an inner loop that calculates the new position for all individuals and examines the probability of substitution. Since finding the new position of the worker and the stone block at the component level is done for each GPC vector, the fundamental operations are performed proportionally to the total number of loops. This is repeated until the end of the algorithm. On the other hand, to define the group of workers, we need $O(n \times Max_{it})$ where n is the number of population and Max_{it} is the maximum iteration of an algorithm. Therefore, if we assume that the algorithm stops at Max_{it} and dim is the number of decision variables, then the GPC time complexity is $O(n \times Max_{it} \times dim)$. The space complexity of the GPC algorithm is the amount of space used during the initialization procedure that is

Table 3 Mean and standard deviation of 50 independent runs in using benchmark functions

Fn	NFE	Algorithms					
		GA	ICA	PSO	DE	FA	**GPC**
f_1	10^3	11.3988 ± 1.4535	14.6126 ± 2.1301	7.6426 ± 0.4695	14.4831 ± 1.2437	17.3214 ± 0.7093	**3.07e−06 ± 3.43e−06**
f_2	10^3	0.3091 ± 0.4750	0.0135 ± 0.0300	0.0774 ± 0.2344	3.03e−09 ± 9.21e−09	0.0001 ± 0.0002	**8.81e−10 ± 1.77e−09**
f_3	10^3	1.1979 ± 2.7912	0.2044 ± 0.6552	0.1256 ± 0.1530	8.57e−09 ± 4.07e−09	0.1361 ± 0.1253	**3.36e−15 ± 2.20e−14**
f_4	10^3	0.3190 ± 0.9152	0.2156 ± 0.9741	0.0002 ± 0.0002	1.35e−10 ± 3.46e−10	0.0002 ± 0.0003	**9.63e−11 ± 2.16e−10**
f_5	10^3	2.4663 ± 6.4898	0.9443 ± 0.1509	0.8296 ± 0.4346	0.2579 ± 0.1520	0.9347 ± 0.3105	**0.1000 ± 4.616e−05**
f_6	10^3	− 1.0258 ± 0.0131	**− 1.0316 ± 4.485e−16**	− 1.0315 ± 3.505e−05	**− 1.0316 ± 4.485e−16**	− 1.0315 ± 4.763e−05	**− 1.0316 ± 4.485e−16**
f_7	10^3	0.0070 ± 0.0166	0.0005 ± 0.0030	1.99e−05 ± 3.11e−05	1.45e−12 ± 5.71e−12	3.16e−05 ± 3.04e−05	**2.97e−20 ± 1.46e−19**
f_8	10^3	− 2.0623 ± 0.0004	**− 2.0626 ± 1.794e−15**	− 2.0625 ± 1.979e−05	**− 2.0626 ± 1.794e−15**	− 2.0625 ± 1.4142	**− 2.0626 ± 1.794e−15**
f_9	10^3	12.6266 ± 11.3721	5.4428 ± 4.4744	5.1396 ± 4.0240	2.6012 ± 2.1694	2.7060 ± 1.6873	**1.5526 ± 1.6249**
f_{10}	10^4	12.8940 ± 4.6453	8.6891 ± 5.1611	1.5832 ± 1.1931	30.9899 ± 40.2824	18.7399 ± 18.2317	**0.6667 ± 0.0001**
f_{11}	10^3	− 0.8655 ± 0.0957	− 0.9289 ± 0.0570	− 0.9890 ± 0.0186	− 0.9714 ± 0.0266	− 0.9644 ± 0.0256	**− 1.0000 ± 0.0000**
f_{12}	10^3	− 0.0428 ± 0.1292	− 0.9795 ± 0.0260	− 0.9857 ± 0.0259	− 0.9767 ± 0.0825	− 0.6120 ± 0.4841	**− 0.9969 ± 0.0061**
f_{13}	10^3	− 705.91 ± 146.12	− 823.73 ± 122.03	− 774.35 ± 131.52	− 876.21 ± 55.486	− 917.69 ± 55.511	**− 923.83 ± 47.916**
f_{14}	10^3	35.9352 ± 16.3258	36.2860 ± 16.9179	8.7594 ± 1.7283	58.1811 ± 15.9275	146.64 ± 22.8454	**0.0647 ± 0.1498**
f_{15}	10^3	− 18.9269 ± 0.5256	− 19.2025 ± 0.0046	− 19.2040 ± 0.0047	− 18.7739 ± 0.9304	− 19.2072 ± 0.0014	**− 19.2076 ± 0.0016**
f_{16}	10^4	48.3567 ± 25.6156	0.9214 ± 2.7513	0.0101 ± 0.0440	29.5942 ± 12.1996	92.5290 ± 17.5702	**2.37e−18 ± 6.23e−18**
f_{17}	10^3	14.0730 ± 4.3324	34.8092 ± 10.6521	7.4679 ± 3.2174	33.4572 ± 9.4299	57.4547 ± 10.5266	**2.7158 ± 0.2025**
f_{18}	10^3	0.0731 ± 0.1709	0.0074 ± 0.0268	0.0007 ± 0.0008	**7.68e−10 ± 3.35e−09**	0.0010 ± 0.0008	0.0023 ± 0.0155
f_{19}	10^3	0.0222 ± 0.0591	0.0036 ± 0.0116	1.71e−05 ± 2.38e−05	8.55e−11 ± 3.67e−10	1.05e−05 ± 1.00e−05	**5.69e−22 ± 1.87e−21**
f_{20}	10^3	− 13.6477 ± 0.9469	− 16.5118 ± 1.4582	− 11.4597 ± 0.8947	− 8.6621 ± 0.5310	− 9.4256 ± 0.6168	**− 5.8100 ± 0.7490**
f_{21}	10^4	4.0745 ± 2.7031	3.4116 ± 3.1863	0.0442 ± 0.0311	32.6352 ± 74.4156	11.3340 ± 11.2445	**4.77e−21 ± 1.12e−20**
f_{22}	10^4	10.8356 ± 3.9949	75.7122 ± 0.6142	51.7387 ± 18.5381	66.6841 ± 36.0603	77.6892 ± 21.6519	**0.0000 ± 0.0000**
f_{23}	10^4	230.83 ± 158.11	170.91 ± 121.87	64.7732 ± 40.2686	105.97 ± 35.6370	240.14 ± 158.57	**28.2754 ± 0.4155**
f_{24}	10^3	0.0528 ± 0.0572	0.0322 ± 0.0512	0.0006 ± 0.0020	0.0017 ± 0.0028	0.0014 ± 0.0031	**0.0000 ± 0.0000**
f_{25}	10^3	0.5001 ± 4.677e−05	0.5001 ± 2.659e−05	0.5000 ± 1.28e−05	0.5000 ± 7.52e−06	0.5000 ± 4.31e−05	**0.5000 ± 1.41e−06**
f_{26}	10^3	− 128.51 ± 47.8553	− 180.39 ± 19.1340	− 186.10 ± 1.2985	− 180.39 ± 6.7993	− 185.21 ± 2.7258	**− 186.71 ± 0.0246**
f_{27}	10^3	12.0705 ± 5.3422	10.1236 ± 4.0827	2.2410 ± 0.4308	18.3657 ± 5.2573	43.3034 ± 7.1993	**2.50e−13 ± 6.25e−13**
f_{28}	10^3	0.0007 ± 0.0015	0.0012 ± 0.0019	9.05e−06 ± 9.21e−06	0.0037 ± 0.0039	0.0244 ± 0.0200	**1.12e−23 ± 6.99e−23**
f_{29}	10^4	0.9914 ± 0.4659	0.0320 ± 0.0972	0.0003 ± 0.0012	15.1790 ± 21.7896	2.2247 ± 0.4904	**6.93e−20 ± 1.70e−19**
f_{30}	10^4	166.96 ± 64.5063	82.4793 ± 27.1642	10.0492 ± 4.4244	131.70 ± 61.5525	115.60 ± 28.1611	**2.16e−19 ± 8.03e−19**

considered at any one time. Thus the space complexity is $O(it \times dim)$, where it is the number of iterations and dim is the number of decision variables as mentioned earlier.

One of the strengths of the GPC algorithm is that the available information of the population is fully utilized and used to intelligently orient the search and optimization process. As a result, better solutions are obtained. The nature of the GPC algorithm is that the previous good information is retained, meaning that the algorithm has memory. This feature does not exist in the GA but similar to this feature seen in the PSO algorithm. In the GA when the population changes, prior knowledge disappears but as mentioned earlier in GPC desirable information is retained. In the proposed algorithm in each iteration, the population is compared with the top member here, the Pharaoh's special agent. In this way, the population actually share their information and knowledge. Similar to this feature is exists in the PSO algorithm. In addition to the good features of the GA and PSO algorithms, the high-speed feature is also inherited from the DE algorithm. DE is a fast and efficient algorithm that uses the direction and distance information of members to find solutions. This feature exists in the GPC algorithm. The convergence speed and speed of finding the global optimum are very high since the entire population is changing their position, this means that each member hopes to find the optimal solution. Worker substitutions are a very useful operation that increases the balance between exploration and exploitation. In Fig. 7, examples of convergence related to some benchmark test functions are shown. In this algorithm, like other population-based algorithms, with an increasing population, we have the flexibility to deal with local minima. The proposed algorithm is also able to work with a very small population (at least 2). As shown in the table of results, the proposed algorithm also can deal with multimodal and nonlinear functions. The low number of parameters is a good feature of the algorithm that makes it

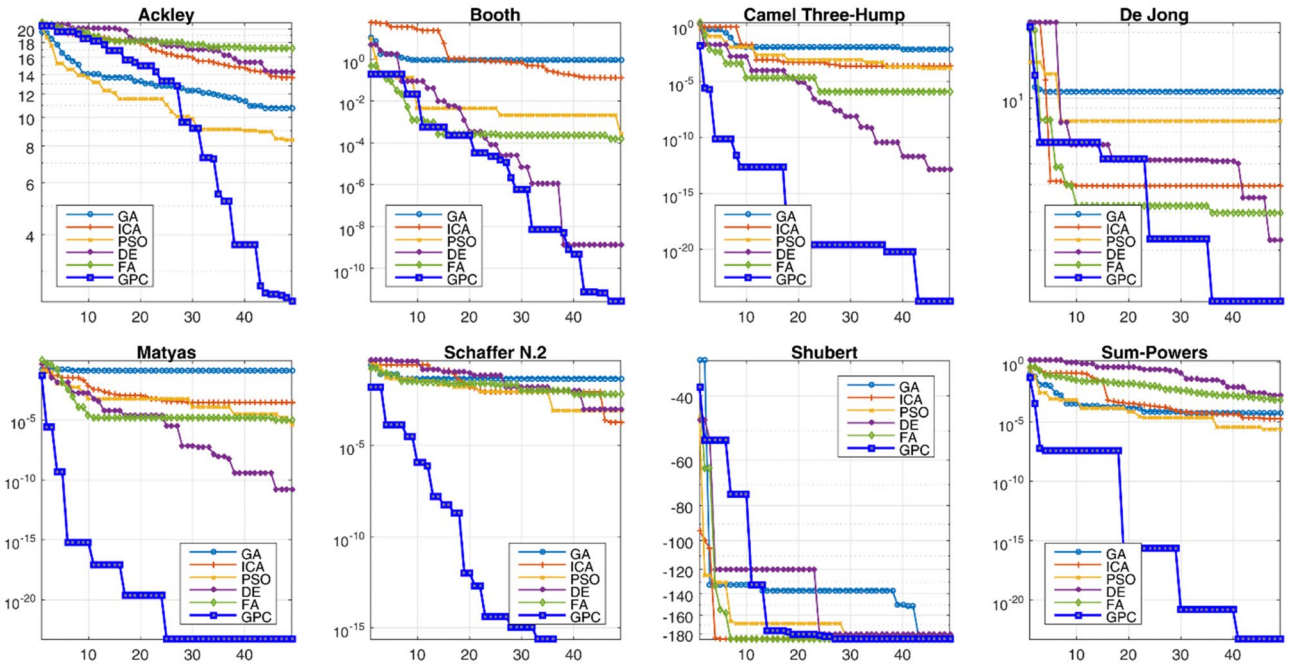

Fig. 7 Comparison of convergence curves of GPC and literature algorithms obtained in some of the benchmark functions

easy to implement. Unlike the PSO, the GPC does not have premature convergence. Figure 8 shows perspective view of the population in consequent iterations for some benchmark functions as an example.

Some of the advantages of the GPC algorithm are described below. The GPC algorithm tries to find the minimum solutions of the fitness function in the set of real numbers. In this algorithm, the population is real vectors. This feature is also present in the DE algorithm. The advantage is that if the objective of optimization with such a fitness function can be formulated, then the use of GPC will be very efficient. It is also better to use GPC in problems that can be solved with evolutionary-based algorithms, because to get acceptable results from evolutionary-based algorithms such as GA, assuming there is no adaptive version, we need to set or tune the parameters correctly, which is time-consuming. In the GPC algorithm, there is the ability to find the global minimum without even considering the initial values for some parameters. It's also very fast and yet simple, so it's easy for implementation. One of the interesting features of this algorithm is the ability to change from multiplicative to the accumulative mode in finding the position of workers, which for some problems leads to more appropriate solutions. With this option, the density of the solutions within the solution space may be much higher than those generated via other metaheuristics. In other words, the solutions can be much closer to each other. This algorithm does not require special control parameters and works well without special control over the parameters. Finally, the substitution probability rate can be controlled. This algorithm has the ability to easily convert from continuous mode to the integer or discrete binary form. There is also the ability to find optimal solutions to a non-linear constrained optimization problem with penalty functions. In addition, due to the fact that all members of the population can try to be the best, there is no problem in getting stuck in the local trap. Another advantage of the algorithm is that if the population increases, the run time will increase linearly. This is while in algorithms such as FA, if the population increases, the run time increases exponentially.

If we want to point out the disadvantages of GPC, it can be said that for some combinatorial problems such as the Traveling Salesman Problem (TSP) may not be appropriate to use unless it is developed. It is better to solve these problems with specific path determination algorithms such as ACO.

5 Statistical analysis

Statistical analysis is the use of statistical data to determine the relationships and probabilities between data quantitatively. This means that the statistical analysis process generates and analyzes statistical data to discover the meaning of the data. The main purpose of statistical analysis is to identify trends.

In this paper, Friedman and Iman-Davenport tests were used to find significant differences between the results of the

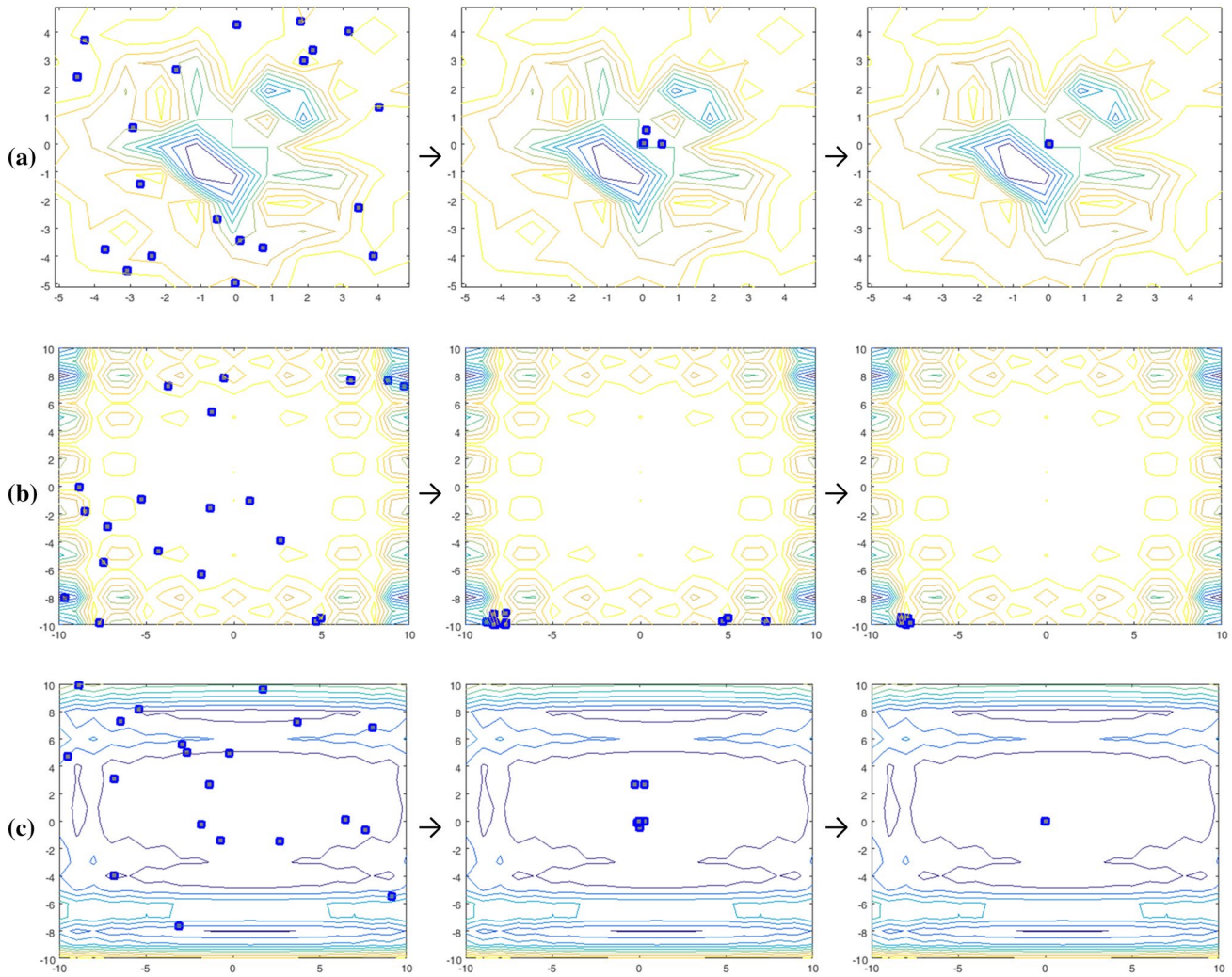

Fig. 8 The perspective view in 1st, 5th, and 10th iteration from left to right, respectively. **a** The drop-wave function. **b** The holder-table function. **c** The Levy function

Table 4 Ranking of the algorithms

	Algorithms					
	GA	ICA	PSO	DE	FA	**GPC**
Ranking	5.05	4.17	2.80	3.50	4.27	**1.22**

Table 5 Results of Friedman's and Iman–Davenport's tests

Test method	Chi square	Degrees of freedom (DF)	*p* value	Hypothesis
Friedman	80.0827	5	8.063e−16	Rejected
Iman–Davenport	31.6940	5	2.200e−16	Rejected

Table 6 Results of the Holm's method based on the mean of 50 independent runs (GPC is the control algorithm)

Algorithm	*j*	α/*j*	*z*-score	*p* value	Hypothesis
PSO	1	0.0500	3.2709	0.001072	Rejected
DE	2	0.0250	4.7200	<0.00001	Rejected
ICA	3	0.0166	6.1070	<0.00001	Rejected
FA	4	0.0125	6.3140	<0.00001	Rejected
GA	5	0.01	7.9288	<0.00001	Rejected

Table 7 The results of applying (mean and standard deviation) the GPC, PSO and DE algorithms on test functions for 5000 and 10,000 dimensions

Fn	NFE	Algorithms					
		PSO		DE		**GPC**	
		dim = 5000	dim = 10,000	dim = 5000	dim = 10,000	dim = 5000	dim = 10,000
f_1	10^3	17.4625 ± 0.4382	17.5663 ± 0.3181	21.2406 ± 0.0068	21.2453 ± 0.0056	**0.0002 ± 0.0001**	**0.0002 ± 9.83e−05**
f_2	10^3	0.1524 ± 0.3213	0.1571 ± 0.3312	0.0258 ± 0.0206	0.0266 ± 0.0343	**0.0215 ± 0.0228**	**0.0210 ± 0.0233**
f_3	10^3	0.1533 ± 0.1833	0.1934 ± 0.1997	0.5515 ± 0.2689	0.6216 ± 0.3458	**0.0000 ± 0.0000**	**0.0000 ± 0.0000**
f_4	10^3	0.5543 ± 0.6417	0.1180 ± 0.3034	0.3446 ± 0.2294	0.2900 ± 0.3844	**0.0005 ± 0.0004**	**0.0002 ± 0.0003**
f_5	10^3	1.1992 ± 0.7266	0.9579 ± 0.5248	1.9607 ± 1.1478	2.2840 ± 1.3836	**0.1000 ± 2.21e−05**	**0.1800 ± 0.2531**
f_6	10^3	− 1.0315 ± 9.6e−05	− 1.0315 ± 0.0001	− 1.0298 ± 0.0011	− 1.0286 ± 0.0033	**− 1.0316 ± 2.34e−16**	**− 1.0316 ± 2.34e−16**
f_7	10^3	1.34e−05 ± 1.7e−05	4.08e−05 ± 4.6e−05	0.0013 ± 0.0019	0.0028 ± 0.0062	**3.15e−21 ± 9.25e−21**	**2.81e−19 ± 8.76e−19**
f_8	10^3	**− 2.0625 ± 3.1e−05**	**− 2.0626 ± 0.0000**	− 2.0624 ± 0.0001	− 2.0624 ± 0.0002	− 2.0624 ± 0.0001	− 2.0625 ± 0.0001
f_9	10^3	7.4621 ± 3.6801	5.1256 ± 3.7951	2.2932 ± 1.9248	2.8160 ± 2.4803	**0.9780 ± 0.0632**	**2.1851 ± 1.7851**
f_{10}	10^4	Infeasible	Infeasible	Infeasible	Infeasible	**0.9998 ± 0.0005**	**0.9096 ± 0.2844**
f_{11}	10^3	− 0.9695 ± 0.0298	− 0.9677 ± 0.0299	− 0.9301 ± 0.0113	− 0.9386 ± 0.0214	**− 1.0000 ± 0.0000**	**− 1.0000 ± 0.0000**
f_{12}	10^3	− 0.9887 ± 0.0123	− 0.9797 ± 0.0360	Infeasible	Infeasible	**− 0.9894 ± 0.0144**	**− 0.9887 ± 0.0140**
f_{13}	10^3	Infeasible	Infeasible	− 905.85 ± 30.481	− 903.60 ± 50.648	**− 932.03 ± 33.306**	**− 911.32 ± 59.638**
f_{14}	10^3	Infeasible	Infeasible	Infeasible	Infeasible	**159.34 ± 160.23**	**302.43 ± 231.76**
f_{15}	10^3	− 18.3868 ± 2.5713	− 18.7544 ± 0.4270	− 18.8650 ± 0.3702	− 19.0553 ± 0.3666	**− 19.1772 ± 0.0273**	**− 19.2031 ± 0.0080**
f_{16}	10^4	Infeasible	Infeasible	Infeasible	Infeasible	**6.87e−12 ± 2.62e−12**	**4.99e−10 ± 4.25e−10**
f_{17}	10^3	12,104 ± 1240	24,421 ± 2681	62,575 ± 671	126,570 ± 708	**454.48 ± 0.040**	**908.72 ± 0.089**
f_{18}	10^3	**0.0005 ± 0.0007**	**0.0019 ± 0.0018**	0.6235 ± 0.5093	0.5282 ± 0.4862	0.0113 ± 0.0092	0.0344 ± 0.0326
f_{19}	10^3	1.05e−05 ± 1.5e−05	1.10e−05 ± 1.4e−05	0.0365 ± 0.0770	0.0295 ± 0.0343	**8.31e−21 ± 2.62e−20**	**3.93e−20 ± 1.22e−19**
f_{20}	10^3	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible	Infeasible
f_{21}	10^4	Infeasible	Infeasible	Infeasible	Infeasible	**4.31e−16 ± 4.14e−16**	**2.13e−15 ± 1.57e−15**
f_{22}	10^4	49,407 ± 998	103,949 ± 1711	Infeasible	Infeasible	**0.0000 ± 0.0000**	**0.0000 ± 0.0000**
f_{23}	10^4	Infeasible	Infeasible	Infeasible	Infeasible	**499.878 ± 0.329**	**999.879 ± 0.325**
f_{24}	10^3	0.0005 ± 0.0010	0.0004 ± 0.0005	0.0435 ± 0.0363	0.0965 ± 0.0557	**0.0000 ± 0.0000**	**0.0000 ± 0.0000**
f_{25}	10^3	0.5001 ± 1.5e−05	0.5001 ± 4.5e−05	0.5001 ± 7.8e−06	0.5001 ± 1.5e−05	**0.5000 ± 6.9e−06**	**0.5000 ± 1.2e−05**
f_{26}	10^3	− 155.04 ± 30.968	− 157.40 ± 32.344	− 157.17 ± 19.980	− 168.52 ± 21.645	**− 183.52 ± 6.5020**	**− 186.27 ± 0.9449**
f_{27}	10^3	7277.5 ± 829	14,849.9 ± 1228	42,690.2 ± 302	86,051.3 ± 258	**1.20e−07 ± 1.17e−07**	**1.29e−07 ± 1.03e−07**
f_{28}	10^3	1.1608 ± 1.9391	5.3391 ± 0.8236	4.0833 ± 0.8357	4.8182 ± 0.6840	**3.22e−23 ± 1.00e−22**	**1.01e−25 ± 1.90e−25**
f_{29}	10^4	Infeasible	Infeasible	Infeasible	Infeasible	**5.01e−13 ± 4.58e−13**	**3.10e−12 ± 4.02e−12**
f_{30}	10^4	Infeasible	Infeasible	Infeasible	Infeasible	**2.52e−07 ± 4.83e−07**	**8.70e−05 ± 0.0002**

GPC algorithm and other algorithms. Table 4 shows Friedman's rankings based on the results of Table 3. As the table shows, the best rank is related to the GPC algorithm and then PSO, DE, ICA, FA, and GA are located, respectively. Table 5 shows the results of the Friedman and Iman-Davenport tests. As Table 5 shows, the hypothesis is rejected according to the results. So we conclude that there is a significant difference in the performance of the algorithms.

Because of the significant difference observed, we use the Holm method as a post hoc test for better analysis. This test compares the best rank obtained from Friedman rankings with the results of other algorithms. Given that the best rank is related to the GPC algorithm, this algorithm is considered as the control algorithm. Also the confidence interval is 95% ($\alpha = 0.05$). The results of the Holm method are outlined in Table 6. The results show that the GPC algorithm has a significant difference in comparison with other algorithms.

6 High-dimensional tests

Over the past decade, high-dimensional problems have received much attention from researchers, and various population-based algorithms have been used to solve these problems. The advances of various sciences and the advances of technology because of the high-dimensionality and larger search space, it has made high-dimensional optimization a major requirement.

One of the things that makes optimization problems difficult is to scale up the search space. Many metaheuristic

Table 8 The image segmentation fitness function results (Mean and Standard Deviation)

Image	K value	Algorithms					
		GA	ICA	PSO	DE	FA	**GPC**
Airplane	K = 3	347,788 ± 25,095	318,821 ± 736	322,467 ± 13,328	318,613 ± 957	318,965 ± 658	**318,080 ± 634**
	K = 4	332,227 ± 41,228	277,192 ± 13,600	274,510 ± 8210	270,625 ± 7	271,031 ± 133	**270,617 ± 26**
	K = 5	295,512 ± 32,007	250,186 ± 9100	246,039 ± 9033	245,117 ± 6232	244,193 ± 2232	**241,317 ± 499**
Baboon	K = 3	978,216 ± 82,200	888,057 ± 16	888,061 ± 14	888,047 ± 18	888,114 ± 34	**888,033 ± 2**
	K = 4	818,485 ± 58,452	724,440 ± 33	724,433 ± 31	735,501 ± 33,705	724,589 ± 59	**724,431 ± 6**
	K = 5	732,511 ± 36,383	653,162 ± 14,549	652,887 ± 14,607	644,639 ± 8881	645,043 ± 8875	**642,019 ± 2990**
Barbara	K = 3	552,524 ± 42,053	504,978 ± 18,815	504,977 ± 18,816	504,978 ± 18,816	496,985 ± 9008	**494,028 ± 4**
	K = 4	486,828 ± 20,584	430,109 ± 18,949	**426,118 ± 16,255**	454,405 ± 95,453	426,469 ± 11,701	429,043 ± 13,076
	K = 5	457,827 ± 23,446	392,425 ± 30,105	375,559 ± 9482	371,760 ± 7152	374,901 ± 12,865	**381,185 ± 25,185**
Boats	K = 3	349,756 ± 38,266	317,820 ± 19,490	310,271 ± 15,913	310,271 ± 15,913	302,827 ± 41	**302,730 ± 8**
	K = 4	317,856 ± 26,618	268,258 ± 1434	269,198 ± 2164	267,849 ± 129	271,964 ± 10,944	**267,777 ± 385**
	K = 5	284,298 ± 17,082	239,904 ± 3700	240,182 ± 4073	241,428 ± 9085	239,876 ± 5723	**239,765 ± 1878**
Butterfly	K = 3	693,739 ± 59,446	600,451 ± 316	600,450 ± 316	600,400 ± 158	600,397 ± 18	**600,351 ± 0.8934**
	K = 4	599,738 ± 62,673	518,572 ± 13,067	510,974 ± 8004	510,976 ± 8003	511,076 ± 8017	**508,475 ± 32**
	K = 5	514,745 ± 24,349	453,833 ± 12,932	450,466 ± 11,307	446,772 ± 10,371	451,457 ± 12,302	**441,812 ± 206**
House	K = 3	546,494 ± 88,219	418,319 ± 76,105	418,319 ± 76,105	400,269 ± 57,079	382,424 ± 88	**382,224 ± 3**
	K = 4	468,701 ± 72,685	336,142 ± 29,247	319,151 ± 27,357	313,499 ± 23,905	**302,746 ± 206**	303,013 ± 1032
	K = 5	416,966 ± 91,710	280,873 ± 7611	**277,556 ± 1793**	285,708 ± 25,697	280,005 ± 5105	286,374 ± 7481
Lena	K = 3	563,491 ± 20,432	541,279 ± 15,811	536,379 ± 316	536,281 ± 7	536,410 ± 187	**536,281 ± 0.8819**
	K = 4	521,361 ± 28,630	473,778 ± 5900	473,183 ± 5115	473,391 ± 5393	470,314 ± 2943	**469,391 ± 411**
	K = 5	464,393 ± 24,779	421,960 ± 8919	417,333 ± 1247	417,160 ± 1334	419,372 ± 4381	**416,199 ± 811**
Peppers	K = 3	868,225 ± 63,115	762,720 ± 632	762,820 ± 674	763,020 ± 1581	762,622 ± 33	**762,524 ± 1**
	K = 4	770,053 ± 59,433	655,089 ± 749	654,830 ± 316	655,433 ± 918	655,146 ± 552	**654,789 ± 17**
	K = 5	701,150 ± 54,943	584,794 ± 21,497	576,263 ± 25,590	573,168 ± 12,238	576,473 ± 16,349	**569,241 ± 3156**

algorithms are not capable of solving high-dimensional problems. Important factors that influence problem-solving with increasing dimensions are as follows. If the size of the problem increases, the search space will increase exponentially. Problem specifications may change with increasing dimensionality, such as Rosenbrock function that is unimodal in two-dimensions and multimodal in more than two-dimensions. Also, evaluating high-dimensional problems is costly. Besides, the interaction between variables is influential. If the variables are independent and the dimensions are increased, we can solve the problem by optimizing each of the variables. But if variables interact, they all have to be optimized together, as a result, optimization becomes difficult with increasing dimensions.

In this section, experiments are performed to evaluate the performance of the proposed algorithm in high dimensions. The difference between the experiments in this section and Sect. 4 is in the number of dimensions and number of independent runs. Here, there are 5000 and 10,000 dimensions. Hence, the mean and standard deviation of 10 independent runs are considered. Also according to Table 4, the proposed GPC algorithm (best ranked), second-best algorithm and third-best algorithm are used for comparison. Therefore, according to the table, the PSO and DE algorithms are selected for comparison. Table 7 shows the results of the execution of the algorithms on the above-mentioned dimensions. Here we point out that some of the solutions obtained from some algorithms are not within the acceptable domain. Therefore, such solutions are specified in the table as "infeasible". This means that the algorithm is not able to provide a proper solution.

As the table shows, the GPC algorithm works well at high dimensions. All the solutions are acceptable and only for the Michalewicz (f_{20}) solutions are not desirable. While the PSO algorithm is unacceptable in 9 functions out of 30 functions. The DE algorithm also fails to perform well in 10 of the 30 benchmark test functions. If we compare the feasible solutions, the solutions provided by the GPC algorithm are much better than the solutions provided by the PSO and DE. This shows that the proposed algorithm is highly capable of solving high-dimensional problems.

Table 9 The CPU consumption time of image segmentation (in second)

Image	K value	Algorithms					
		GA	ICA	PSO	DE	FA	**GPC**
Airplane	K = 3	18.615	19.131	18.592	**18.220**	18.386	18.443
	K = 4	21.366	21.595	21.813	21.462	21.386	**21.164**
	K = 5	24.375	24.436	24.456	24.411	24.891	**24.364**
Baboon	K = 3	18.427	18.208	18.258	18.637	19.107	**18.203**
	K = 4	**21.419**	21.776	21.507	21.487	21.866	21.794
	K = 5	24.265	24.335	24.324	23.968	24.008	**23.683**
Barbara	K = 3	18.778	18.920	18.755	18.689	19.057	**18.520**
	K = 4	21.578	21.872	21.686	21.652	21.855	**21.117**
	K = 5	24.679	24.539	24.485	24.571	24.751	**24.164**
Boats	K = 3	18.676	19.084	19.571	20.009	20.313	**18.364**
	K = 4	21.615	22.092	22.662	24.658	22.907	**21.596**
	K = 5	25.240	25.035	26.572	26.288	26.466	**24.512**
Butterfly	K = 3	**18.265**	19.034	18.274	19.398	19.242	18.441
	K = 4	21.504	21.602	21.412	21.640	21.468	**21.229**
	K = 5	24.439	24.268	24.210	25.517	24.679	**24.076**
House	K = 3	18.664	18.859	18.868	18.919	18.356	**18.257**
	K = 4	21.703	21.145	21.767	21.175	21.452	**21.006**
	K = 5	24.587	24.232	26.658	24.466	24.615	**24.155**
Lena	K = 3	18.490	19.642	21.365	19.706	19.290	**18.332**
	K = 4	21.686	22.039	22.509	21.571	21.484	**21.417**
	K = 5	24.736	25.833	25.367	**23.437**	24.554	24.593
Peppers	K = 3	19.289	21.289	19.649	19.415	19.958	**18.308**
	K = 4	22.032	23.486	23.517	22.746	22.756	**21.779**
	K = 5	24.631	24.555	27.499	25.262	25.353	**24.190**

7 Application in image segmentation

Digital images comprise a large part of scientific studies. In the meantime, image segmentation is one of the activities that is done on digital images. Image segmentation is the process of dividing an image into homogeneous areas [59]. The homogeneous area contains the desired object or part of the object. Homogeneous areas can be determined and measured using some image features such as pixel intensity, color, texture, shape, size and so on [60]. The segmentation is done so that each of the pixels in the specified region has similar properties but in comparison with the adjacent pixels in the adjacent region have different properties. Segmentation can be seen as a key step in preparing the image for applications such as machine vision, and image analysis including medical images, satellite images, and so on.

There are generally four ways of image segmentation. These methods are threshold-histogram-based methods, texture-based methods, split-merge-based methods, and clustering-based methods [61]. Given that in image segmentation, the image is divided into segments where each segment contains a color spectrum or part of a color spectrum, so segmentation can be considered as a clustering problem. Clustering is an unsupervised machine learning technique in which the cluster members should be most similar to each other and also most distinct from the other clusters.

Many clustering algorithms are widely used to solve image segmentation problems. One of the algorithms used in segmentation is the k-means algorithm [62]. The k-means needs to determine the centroids of the clusters, so by determining the centroids and checking the distance of the elements from the centroids, the cluster is specified. The k-means is simple but has a major drawback [63]. Determining the optimal centroid is an NP-hard optimization problem which is quite effective in the quality of the final solutions. In k-means, the initial centroids are randomly selected, while the optimal centroids can be determined by specific approaches. One of these approaches is to use optimization algorithms. The process of optimal centroid search can be considered as an optimization problem. In image segmentation, optimizing the time consumed by the CPU is also important. We know that the image segmentation can be considered as a high-dimensional problem. Using fast and accurate metaheuristic algorithms can be an effective approach. Therefore in this paper, as an application, the proposed GPC optimization algorithm is employed for image clustering.

The procedure and details of this experiment are as follows: the algorithms selected for the experiment were

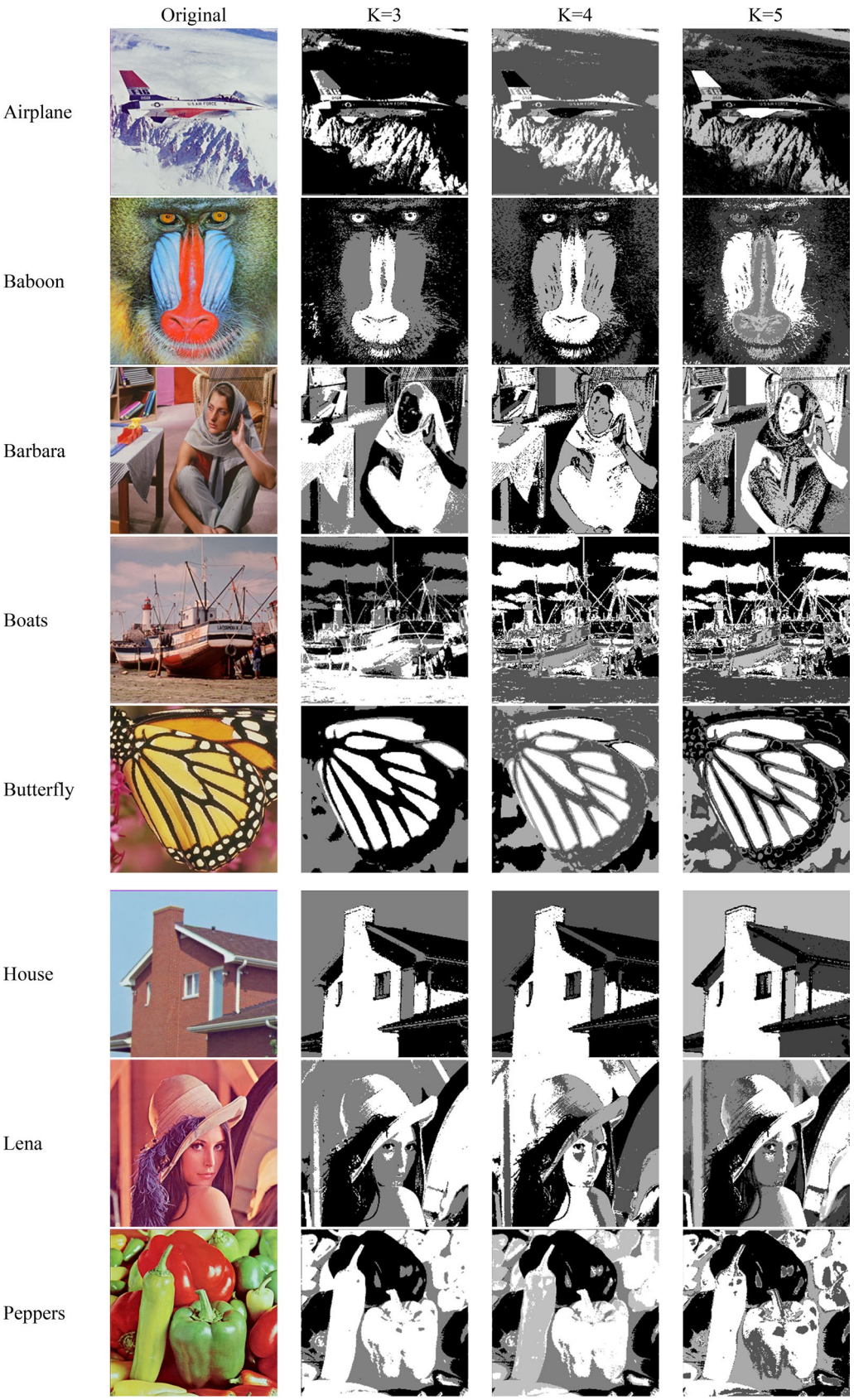

Fig. 9 Results of applying the GPC algorithm to images

applied to eight 256×256 color images. For this purpose, a fitness function is used that measures the within-cluster distances. So, as a fitness function, we have,

$$\text{within cluster distance} = \sum_{i} \sum_{x \in c_i} d(x, m_i) \tag{10}$$

where d is the Euclidean distance, c is the cluster, i specifies the cluster number, x is a member of the cluster, and m is the centroid of the cluster. The purpose is to minimize the above fitness function. If it is minimized, it means that the most optimal centroids are selected. The algorithms are applied to the fitness function and the mean and standard deviation of 10 independent runs are recorded. Also, the average CPU consumption is recorded as an evaluation criterion. The stop condition is 5000 NFE calls for all algorithms. We also performed experiments for K equal to 3, 4, and 5. Table 8 shows the fitness function results for all eight images. Also, Table 9 shows the CPU consumption time for all algorithm. Furthermore, Fig. 9 shows the results of applying the GPC algorithm to images.

Table 8 shows that the proposed algorithm works well for the image segmentation problem, which is a high-dimensional problem. As can be seen, the lowest value of the fitness function in most experiments on different images is provided by the GPC algorithm. Also, in terms of CPU consumption, which can be considered as an evaluation criterion, the proposed algorithm consumed the least time among the other algorithms. This shows that the proposed algorithm is fast and efficient.

8 Conclusions

In this paper, a new metaheuristic algorithm based on the construction method of Giza Pyramids in the ancient past called Giza Pyramids Construction (GPC) was proposed. The proposed algorithm is a population-based algorithm that is controlled by the movements of the workers and pushing the stone blocks on the ramp. This algorithm can be used as a fundamental optimization method in many areas of knowledge including engineering sciences. Experiments have shown that the proposed algorithm is capable of dealing with high-dimensional problems. Evaluations also have shown that the proposed algorithm works well for use in image processing and segmentation. In this paper, in addition to presenting a new algorithm, a new source of inspiration is introduced. The limitations and the lack of complexity in the ancient past have led to many optimal methods. Inspired by ancient past methods, we can be reached to new approaches for solving various optimization problems.

As future work to expand this source of inspiration, studies on the technologies of construction of other ancient civilizations can be made. Other applications of the proposed algorithm can also be explored in a variety of issues such as engineering problems.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Rao SS (2019) Engineering optimization: theory and practice. Wiley, Hoboken
2. Hussain K, Salleh MNM, Cheng S, Shi Y (2019) Metaheuristic research: a comprehensive survey. Artif Intell Rev 52(4):2191–2233
3. Abdel-Basset M, Abdel-Fatah L, Sangaiah AK (2018) Metaheuristic algorithms: a comprehensive review. In: Sangaiah AK, Sheng M, Zhang Z (eds) Computational intelligence for multimedia big data on the cloud with engineering applications. Academic Press, Cambridge, pp 185–231
4. Karkalos NE, Markopoulos AP, Davim JP (2019) Evolutionary-based methods. In: Karkalos NE, Markopoulos AP, Davim JP (eds) Computational methods for application in industry 4.0. Springer, Cham, pp 11–31
5. Holland JH (1992) Genetic algorithms. Sci Am 267(1):66–73
6. Neri F, Cotta C, Moscato P (2011) Handbook of memetic algorithms, vol 379. Springer, Berlin
7. Das S, Suganthan PN (2010) Differential evolution: a survey of the state-of-the-art. IEEE Trans Evol Comput 15(1):4–31
8. Geem ZW, Kim JH, Loganathan GV (2001) A new heuristic optimization algorithm: harmony search. Simulation 76(2):60–68
9. De Castro LN, Von Zuben FJ (2000) The clonal selection algorithm with engineering applications. In: Proceedings of GECCO, vol 2000, pp 36–39
10. Glover F, Laguna M (1998) Tabu search. In: Du DZ, Pardalos PM (eds) Handbook of combinatorial optimization. Springer, Boston, pp 2093–2229
11. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680
12. Lourenço HR, Martin OC, Stützle T (2003) Iterated local search. In: Glover F, Kochenberger GA (eds) Handbook of metaheuristics. International Series in Operations Research & Management Science, vol 57. Springer, Boston, pp 320–353
13. Voudouris C, Tsang EP, Alsheddy A (2010) Guided local search. In: Gendreau M, Potvin JY (eds) Handbook of metaheuristics. International Series in Operations Research & Management Science, vol 146. Springer, Boston, pp 321–361
14. Feo TA, Resende MG (1995) Greedy randomized adaptive search procedures. J Glob Optim 6(2):109–133
15. Hansen P, Mladenović N (2001) Variable neighborhood search: principles and applications. Eur J Oper Res 130(3):449–467
16. Dhal KG, Ray S, Das A, Das S (2019) A survey on nature-inspired optimization algorithms and their application in image enhancement domain. Arch Comput Methods Eng 26(5):1607–1638
17. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of ICNN'95-international conference on neural networks, vol 4. IEEE, pp 1942–1948
18. Yang XS (2009) Firefly algorithms for multimodal optimization. In: International symposium on stochastic algorithms. Springer, Berlin, pp 169–178

19. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Glob Optim 39(3):459–471
20. Dorigo M, Birattari M, Stutzle T (2006) Ant colony optimization. IEEE Comput Intell Mag 1(4):28–39
21. Harifi S, Khalilian M, Mohammadzadeh J, Ebrahimnejad S (2019) Emperor Penguins Colony: a new metaheuristic algorithm for optimization. Evol Intell 12(2):211–226
22. Gandomi AH, Alavi AH (2012) Krill herd: a new bio-inspired optimization algorithm. Commun Nonlinear Sci Numer Simul 17(12):4831–4845
23. Askarzadeh A (2016) A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm. Comput Struct 169:1–12
24. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
25. Jain M, Maurya S, Rani A, Singh V (2018) Owl search algorithm: a novel nature-inspired heuristic paradigm for global optimization. J Intell Fuzzy Syst 34(3):1573–1582
26. Abualigah L, Shehab M, Alshinwan M, Mirjalili S, Abd Elaziz M (2020) Ant lion optimizer: a comprehensive survey of its variants and applications. Arch Comput Methods Eng. https://doi.org/10.1007/s11831-020-09420-6
27. Lam AY, Li VO (2009) Chemical-reaction-inspired metaheuristic for optimization. IEEE Trans Evol Comput 14(3):381–399
28. Hatamlou A (2013) Black hole: a new heuristic optimization approach for data clustering. Inf Sci 222:175–184
29. Abualigah L (2020) Multi-verse optimizer algorithm: a comprehensive survey of its results, variants, and applications. Neural Comput Appl. https://doi.org/10.1007/s00521-020-04839-1
30. Kaveh A, Dadras A (2017) A novel meta-heuristic optimization algorithm: thermal exchange optimization. Adv Eng Softw 110:69–84
31. Atashpaz-Gargari E, Lucas C (2007) Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In IEEE congress on evolutionary computation. IEEE, pp 4661–4667
32. Reynolds RG (1994) An introduction to cultural algorithms. In: Proceedings of the third annual conference on evolutionary programming. World Scientific, River Edge, pp 131–139
33. Rao RV, Savsani VJ, Vakharia DP (2011) Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput Aided Des 43(3):303–315
34. Askari Q, Younas I, Saeed M (2020) Political optimizer: a novel socio-inspired meta-heuristic for global optimization. Knowl Based Syst 195:105709
35. Mehrabian AR, Lucas C (2006) A novel numerical optimization algorithm inspired from weed colonization. Ecol Inform 1(4):355–366
36. Ma L, Zhu Y, Liu Y, Tian L, Chen H (2015) A novel bionic algorithm inspired by plant root foraging behaviors. Appl Soft Comput 37:95–113
37. Rezaei N, Ebrahimnejad S, Moosavi A, Nikfarjam A (2019) A green vehicle routing problem with time windows considering the heterogeneous fleet of vehicles: two metaheuristic algorithms. Eur J Ind Eng 13(4):507–535
38. Bekdaş G, Nigdeli SM, Kayabekir AE, Yang XS (2019) Optimization in civil engineering and metaheuristic algorithms: a review of state-of-the-art developments. In: Computational intelligence, optimization and inverse problems with applications in engineering. Springer, Cham, pp 111–137
39. Harifi S, Khalilian M, Mohammadzadeh J, Ebrahimnejad S (2020) Optimizing a neuro-fuzzy system based on nature inspired emperor penguins colony optimization algorithm. IEEE Trans Fuzzy Syst 28(6):1110–1124
40. Ghosh M, Guha R, Singh PK, Bhateja V, Sarkar R (2019) A histogram based fuzzy ensemble technique for feature selection. Evol Intell 12(4):713–724
41. Abualigah LM, Khader AT, Hanandeh ES (2018) A new feature selection method to improve the document clustering using particle swarm optimization algorithm. J Comput Sci 25:456–466
42. Abualigah LMQ, Hanandeh ES (2015) Applying genetic algorithms to information retrieval using vector space model. Int J Comput Sci Eng Appl 5(1):19
43. Finley MI (1985) Ancient history, evidence and models. Chatto & Windus, London
44. Momigliano A (1950) Ancient history and the antiquarian. J Warbg Court Inst 13(3/4):285–315
45. Spaulding AC (2017) Explanation in archeology. In: Binford L (ed) Archeology in cultural systems. Routledge, Abingdon, pp 33–39
46. Gates C (2011) Ancient cities: the archaeology of urban life in the ancient Near East and Egypt, Greece and Rome. Taylor & Francis, Abingdon
47. Laurence R (2004) The uneasy dialogue between ancient history and archaeology. In: Sauer E (ed) Archaeology and ancient history. Routledge, Abingdon, pp 111–125
48. Verboven K (2014) Attitudes to work and workers in classical Greece and Greece and Rome. Tijdschrift voor Economische en Sociale Geschiedenis 11:67–87
49. Noorbergen R (2001) Secrets of the lost races: new discoveries of advanced technology in ancient civilizations. TEACH Services Inc., Fort Oglethorpe
50. Flohr M (2015) Innovation and society in the Roman World. Oxford Handbooks Online
51. Verner M (2007) The pyramids: the mystery, culture, and science of Egypt's great monuments. Open Road+Grove/Atlantic
52. Magli G (2009) Akhet Khufu: archaeo-astronomical hints at a common project of the two main pyramids of Giza, Egypt. Nexus Netw J 11(1):35–50
53. Morishima K, Kuno M, Nishio A, Kitagawa N, Manabe Y, Moto M et al (2017) Discovery of a big void in Khufu's Pyramid by observation of cosmic-ray muons. Nature 552(7685):386–390
54. Lehner M (1997) The complete pyramids. Thames & Hudson, London
55. Smith CB (1999) Program management BC. Civ Eng 69(6):34
56. Smith CB (2018) How the great pyramid was built. Smithsonian Institution, Washington, DC
57. Rigby JK (2016) Building the great pyramid at Giza: investigating ramp models
58. Surjanovic S, Bingham D (2013) Virtual library of simulation experiments: test functions and datasets. Retrieved March 4, 2020, from http://www.sfu.ca/~ssurjano
59. He L, Huang S (2017) Modified firefly algorithm based multilevel thresholding for color image segmentation. Neurocomputing 240:152–174
60. Pare S, Kumar A, Bajaj V, Singh GK (2016) A multilevel color image segmentation technique based on cuckoo search algorithm and energy curve. Appl Soft Comput 47:76–102
61. Jia H, Lang C, Oliva D, Song W, Peng X (2019) Hybrid grasshopper optimization algorithm and differential evolution for multilevel satellite image segmentation. Remote Sens 11(9):1134
62. Kapoor S, Zeya I, Singhal C, Nanda SJ (2017) A grey wolf optimizer based automatic clustering algorithm for satellite image segmentation. Procedia Comput Sci 115:415–422
63. Hrosik RC, Tuba E, Dolicanin E, Jovanovic R, Tuba M (2019) Brain image segmentation based on firefly algorithm combined with k-means clustering. Stud Inform Control 28:167–176