An efficient Planet Optimization Algorithm for solving engineering problems

Thahn Sang-To1,2, Minh Hoang-Le1,2, Magd Abdel Wahab3,4,5 & Thanh Cuong-Le2,6,7

In this study, a meta-heuristic algorithm, named The Planet Optimization Algorithm (POA), inspired by Newton’s gravitational law is proposed. POA simulates the motion of planets in the solar system. The Sun plays the key role in the algorithm as at the heart of search space. Two main phases, local and global search, are adopted for increasing accuracy and expanding searching space simultaneously. A Gauss distribution function is employed as a technique to enhance the accuracy of this algorithm. POA is evaluated using 23 well-known test functions, 38 IEEE CEC benchmark test functions (CEC 2017, CEC 2019) and three real engineering problems. The statistical results of the benchmark functions show that POA can provide very competitive and promising results. Not only does POA require a relatively short computational time for solving problems, but also it shows superior accuracy in terms of exploiting the optimum.

In recent years, many nature-inspired optimization algorithms have been proposed. Some of swarm-inspired algorithms are appreciated such as Particle Swarm Optimization algorithm (PSO)1, Firefly Algorithm (FA)2, Dragonfly Algorithm (DA)3, Whale Optimization Algorithm (WOA)4, Grey Wolf Optimizer (GWO)5, Monarch Butterfly Optimization (MBO)6, Earthworm Optimization Algorithm (EWA)7, elephant herding optimization (EHO)8, moth search (MS) algorithm9, Slime Mould Algorithm (SMA)10, Colony Predation Algorithm (CPA)10 and Harris Hawks Optimization (HHO)11. Besides, quite a number of physics-inspired algorithm simulated physical laws in the universe or nature, such as Curved Space Optimization (CSO)12, Water Wave Optimization (WWO)13, etc. Moreover, some algorithms based on the mathematical foundations are also creative approaches, e.g. Runge Kutta optimizer (RUN)14.

On the other hand, some algorithms simulate human behavior such as Teaching–Learning-Based Optimization (TLBO)15, and Human Behavior-Based Optimization (HBBO)16. Meanwhile, Genetic Algorithm (GA)17 is inspired by evolution, and achieves a lot of success in solving optimization problems in many fields. With the growing popularity of GA, many evolutions-based algorithms are proposed in the literature, including Evolutionary Programming (EP)18, and Evolutionary Strategies (ES)19.

Nowadays, metaheuristic algorithms become an essential tool for solving complex optimization problems in various fields. Many researchers applied such algorithms to make an effort to deal with difficult issues in biology20, economics21, engineering22,23, etc. Therefore, constructing new algorithms to meet such complex requirements has a significant merit.

In this paper, we propose an optimization algorithm using Newton’s law of universal gravitation as the basis for its development. In this algorithm, a number of pre-eminent features are considered, such as local search, global search, to increase the ability for finding the exact solutions built into simulating the planets’ movement in the universe.

This research paper is structured into several sections as follows. In the next section, the construction of a meta-heuristic algorithm is presented. The structural POA is simulated based on Newton’s law of universal
gravitation and astronomical phenomena. Then, a wide range of applications of various benchmark problems is used to demonstrate how effective POA is. At the same time, we present the applications of the POA to real engineering problems. Finally, based on the results presented, the last section reports the conclusions.

The Planet Optimization Algorithm (POA)

Physics is a fundamental science whose laws governs everything from the tiniest object electrons, neutrons, or protons to extremely massive stars or galaxies (about a hundred thousand light-years across). The laws of physics are widely applied in everyday life from transportation to medicine, from agriculture to industry, etc. In science, it is also the foundation for many other sciences such as chemistry, biology, even math. In the field of artificial intelligence (AI), the laws of physics are the inspiration for many optimization algorithms. In the study, we also present an algorithm based on such a physical law.

Inspiration.

Inspired by the laws of the motion in the universe, an algorithm is proposed from the interaction of mutual gravitational between the planets. Specifically, this optimization algorithm simulates the universal gravitation laws of Isaac Newton. The core of this algorithm is given as follows:

\[
-\vec{F} = G \times \frac{m_1 \times m_2}{R^2}
\]  

(1)

where \(\vec{F}\): The gravitational force acting between two planets; \(G\): The gravitational constant; \(R\): The distance between two planets; \(m_1, m_2\): The mass of the two planets.

The gravitation of a two-planet (as shown in Fig. 1) is dependent on as Eq. (1). However, in this study, we find that the value of force \(\vec{F}\) will give less effective results than when using the moment \((M)\) as a parameter in the search process of the algorithm.

\[
|\vec{M}| = |\vec{F}| \times R
\]  

(2)

The planet optimization algorithm.

The universe is infinitely big and has no boundary, and it is a giant space that is filled with galaxies, stars, planets, and many and many interesting astrophysical objects. For simplicity and ease of visualization, we use the solar system to make representation for this algorithm simulation.

First of all, a system that consists of the Sun, the Earth and the Moon (as shown in Fig. 2) is considered in this case. Of course, everybody understands that the Sun maintains its gravitation to keep the Earth moving around it. Interestingly, the mass of the Sun is 330,000 times higher than that of the Earth. However, the Earth also creates a gravitational force large enough to keep the Moon in orbit around the Earth. This demonstrates that two
factors influence the motion of a planet, not only the mass but also the distance between the two planets. An algorithm simulating the law of universal gravitation is, therefore, presented as follows:

- The Sun will act as the best solution. In the search space, it will have the greatest mass, which means it will have a greater gravitational moment for the planets around and near it.
- Between the Sun other planets, there is a gravitational attraction moment between each other. However, this moment depends on the mass as well as the distance between these two objectives. This means that, although the Sun has the largest mass compared to other planets, its moment on the too distant planets is negligible. This helps the algorithm to avoid local optimization as illustrated in Fig. 3.

In the \( t \)th iteration, the mass of red planet (see Fig. 3) is the biggest, so it represents the Sun. As the pink planets are close to the Sun, they will move to the location of the Sun because of a gravitational attraction moment (\( M_{t}^{p} \)) between the Sun and the planets. Nevertheless, the red planet (or the Sun) in the \( t \)th iteration does not have the desired position that we are looking for, i.e. a minimum optimum. In other words, if all planets move to the red planet, the algorithm is stuck in the local space. In contrast, the blue planet is a potential location and far from the Sun. The interaction of the Sun with the blue planet (\( M_{t}^{b} \)) is small, because it is far from the Sun in the \( t \)th iteration. Thus, it is quite free for the blue planet to search a better location in the next iterations.

The main core of the algorithm is based on the above 2 principles. Besides, the Sun is the true target of searching, and of course we don’t have its exact location. In this case, the planet with the highest mass in the \( t \)th iteration would act as the Sun at the same time.

The implementation of the algorithm is as follows:

**Stage 1: the best start.** Ideally, a good algorithm is the one in which the final best solution should be independent of the initial positions. Nevertheless, the reality is exactly the opposite for almost all stochastic algorithms. If the objective region is hilly and the global optimum is located in an isolated minor area, an initial population has an important role. If an initial random population does not create any solution in the vicinity of the global search level of the original population, the probability that the population concentrates on true optimum can be very low.

In contrast, with building initial solutions near the global optimal position, the probability of the convergence of the population to the optimal location is very high. Globalization is indeed very high, and consequently, population initialization plays a vital role. Ideally, the initiation should use the critical sampling method, such as techniques applied to the Monte Carlo method in order to sample the solutions for an objective context. This, however, requests enough intellect of the problem and cannot be satisfied for most algorithms.

Similar to choosing initial population, choosing the best solution in the original population to the role of the Sun with respect to all the other planets moving to the position is important. This selection will determine the convergence speed as well as the accuracy of the algorithm in the future.

Therefore, the algorithm’s first step is to find an effective solution to play a role of the best solution to increase the convergence and accuracy of the search problem in the first iterations.

**Stages 2: M factor.**

\[
M = |\mathbf{F}^{i}| R_{ij} = G \frac{m_{i} m_{j}}{R_{ij}^{2}} \times R_{ij}
\]

In Eq. (3), the following parameters are defined:
• The mass of the planets:

\[ m_i, m_j = \frac{1}{a^{obj_{i,j}/a}} \]

where \( a = 2 \) is a constant parameter, and \( \alpha = [\max(obj) - obj_{sun}] \). This means that if the objective function value of a planet is smaller, the mass of this planet is larger. \( obj_{i,j}, \max(obj), obj_{sun} \) are the values of objective function of the \( i \)th or \( j \)th planet, the worst planet and the Sun, respectively.

• The distance between any 2 objects \( i \) and \( j \) with "Dim" as dimensions, Cartesian distance, is calculated by Eq. (5):

\[ R_{ij} = \left\| X_i - X_j \right\| = \sqrt{\sum_{k=1}^{\text{Dim}} (X_i^k - X_j^k)^2} \]  

(5)

• \( G \) is a parameter, and it is equal to unity in this algorithm.

Stage 3: Global search. From the above, a formula built to simulate global search is indicated by Eq. (6)

\[ X_i^{t+1} = X_i^t + b \times \beta \times r_1 \times (\frac{M_i^t}{M_{\text{Sun}}^t} - X_{\text{Sun}}^t) \]

(6)

The lefthand side of the formula illustrates the current position of a planet \( i \)th in the \( (t+1) \) iteration, while the righthand side consists of the main elements as follows:

• \( X_i^{t+1} \) is the current position of a planet \( i \)th in the iteration \( t \)th.
• \( \beta \approx \frac{M_i^t}{M_{\text{Sun}}^t} \), \( r_1 \approx \text{rand}(0, 1), b \) is a constant parameter.
• \( X_{\text{Sun}}^t \) is the current position of the Sun in the iteration \( t \)th.

where \( \beta \) is a coefficient that depends on \( M \), as shown in Eq. (3), in which \( M_i^t \) is the Sun’s gravity on a planet \( i \)th at \( t \) iteration, and \( M_{\text{max}}^t \) is the value of \( \max(M_i^t) \) at \( t \) iteration. Therefore, the \( \beta \) coefficient contains values in the interval \((0, 1)\).

Stage 4: Local search. In the search process, the true location is always the desired target to be found. However, this goal will be difficult or easy to achieve in this process depending on the complexity of the problem. In most cases, it is only possible to find an approximate value that fits the original requirement. That is to say, the true Sun location yet is in the space between the found solutions.

Interestingly, although Jupiter is the most massive planet in the solar system, Mercury is the planet, for which its location is the closest the Sun. It means that the best solution position to true Sun location at the \( t \) iteration may not be closer than the location of some other solutions to the true location of the Sun.

When the distance between the Sun and planets is small, the local search process is run. As mentioned above, the planet with the biggest mass will operate as the Sun, and in that case, it is Jupiter. Planets near the Sun will go to the location of the Sun. In other words, the planets move a small distance between it and the Sun at \( t \) iteration instead of going straight towards the Sun. The aim of this step is to increase accuracy in a narrow area of search space. Eq. (7) indicates the process for local search as follows:

\[ X_i^{t+1} = X_i^t + \epsilon \times r_2 \times (r_2 \times X_{\text{Sun}}^t - X_i^t) \]

(7)

where \( \epsilon = c_0 - t/T, t \) is the \( t \)th iteration, \( T \) is the maximum number of iterations, and \( c_0 = 2, r_2 \) is Gauss distribution function illustrated by Eq. (8).

\[ f(x, \mu, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} \exp \left( -\frac{(x - \mu)^2}{2\sigma^2} \right) \]

(8)

Many evolutionary algorithms are also randomized by applying common stochastic processes such as power-law distribution and Lévy distribution. However, Gaussian distribution or normal distribution is the most popular since the large number of physical variables (see Fig. 4), including light intensity, errors/uncertainty in measurements, and many other processes, obey this distribution.

The coefficient \( r_2 \) is the Gaussian distribution with mean value \( \mu = 0.5 \) and standard deviation \( \sigma = 0.2 \). It means that 68.2% of \( r_2 \) is in zone 1 about \( (\mu - \sigma) = 0.3 \) to \( (\mu + \sigma) = 0.7 \), and 27.2% of its values is in zone 2 from \( (\mu - 2\sigma) \) to \( (\mu + 2\sigma) \). In other words, POA will move to around the Sun without ignoring potential solutions in local search.

Exploitation employs any data obtained from the issue of interest to create new solutions, which are better than existing solutions. This process and information (for instance gradient), however, are normally local. Therefore, this search procedure is local. The result of search process typically leads to high convergence rates, and it is the strong point of exploitation (or local search). Nevertheless, the weakness of local search is that normally it gets stuck in a local mode.
In contrast, exploration is able to effectively explore the search space, and it typically creates many diverse solutions far from the current solutions. Thus, exploration (or global search) is normally on a global scale. The great strength of global search is that it rarely gets stuck in a local space. The weakness of the global search, however, is slow convergence rates. Besides, in many cases, it wastes effort and time since a lot of new solutions can be far from the global solution.

Figure 5 shows the operation of this algorithm, in which two local and global search processes are governed by the distance parameter \( R_{\text{min}} \). This means that a planet far away from the Sun will be moved depending on Newton law. In contrast, for planets very close to the Sun, the effect of the newton force is so great. They are only moving, therefore, around the Sun. A planet, which is close to the Sun, will support the Sun in exploring a local

| Create a initial population of planets (N planets) |
|--------------------------------------------------|
| Create parameters \( b, c, G, R_{\text{min}} \)    |
| Calculate the fitness of each planet             |
| \( X_{\text{new}} \) = the best planet          |
| Search max(fitness)                              |
| While \( t < T \) (Max number of iterations)     |
| Update \( c \) parameter                         |
| Calculate moment force (\( M \)) of each planet |
| Calculate distance between sun and each          |
| planet \( R_{\text{sun-planet}(i)} \)           |
| Calculate the mass of each planet                |
| Search \( M_{\text{max}} \)                       |
| For \( i = 1 \) to \( N \)                      |
| \( R_{\text{sun-planet}(i)} > R_{\text{min}} \) |
| Local Search                                     |
| Calculate \( r_1, r_2 \)                         |
| \( X_i^{t+1} = X_i^t + c \times r_1 \times (r_2 \times X_{\text{new}}^t - X_i^t) \) |
| True                                            |
| False                                           |
| Global Search                                    |
| Calculate \( r_1, \beta \)                      |
| \( X_i^{t+1} = X_i^t + \beta \times r_1 \times (X_{\text{new}}^t - X_i^t) \) |
| Return best solution                             |
| Stop condition                                   |

Figure 5. Flow chart of the proposed POA.

In contrast, exploration is able to effectively explore the search space, and it typically creates many diverse solutions far from the current solutions. Thus, exploration (or global search) is normally on a global scale. The great strength of global search is that it rarely gets stuck in a local space. The weakness of the global search, however, is slow convergence rates. Besides, in many cases, it wastes effort and time since a lot of new solutions can be far from the global solution.

Figure 5 shows the operation of this algorithm, in which two local and global search processes are governed by the distance parameter \( R_{\text{min}} \). This means that a planet far away from the Sun will be moved depending on Newton law. In contrast, for planets very close to the Sun, the effect of the newton force is so great. They are only moving, therefore, around the Sun. A planet, which is close to the Sun, will support the Sun in exploring a local...
search space, as shown in Eq. (7), while the motion of distant planets from the Sun is less affected by this star at the same time. It means they have a chance to find new potential stars. Search local and global spaces runs simultaneously. This guarantees the enhancement of the accuracy of the search process, but this algorithm does not miss the potential locations.

The parameter $R_{\text{min}}$ must satisfy the following two conditions:

- If the $R_{\text{min}}$ is too large, the algorithm will focus on local search in the first iterations. Therefore, the probability of finding a potential location far away from the present one is difficult.
- In contrast, if $R_{\text{min}}$ is too small, the algorithm focuses on global search. In other words, the exploration of POA in the zone around the Sun is not thorough. Consequently, the best value of the search process may not satisfy the condition.

$$R_{\text{min}} = \frac{\left( \sum_{i=1}^{\text{Dim}} (\text{up}_i - \text{low}_i)^2 \right) / R_0}{R_{\text{min}}}$$  \hspace{1cm} (9)

In this study, $R_{\text{min}}$ is chosen by dividing the search space into 1000 ($R_0 = 1000$) zones. Where 'low' and 'up' are lower and upper bounds of each problem, respectively. With an explicit structure consisting of 2 local and global search processes, POA has satisfied the above two issues and promises to be effective and saving of time in solving complex problems.

Results and discussion

In this section, POA is compared with a series of algorithms using well-known problems. The investigations run on the operating system of Windows 11th Gen Intel(R) Core(TM) i7-1185G7 @ 3.00 GHz 1.80 GHz with RAM 16.0 GB.

Experimental results using classical benchmark functions. In this subsection, POA is employed to handle a wide range of applications of various benchmark problems. A set of mathematical functions with known global optima is commonly employed to validate the effectiveness of the algorithms. The same process is also followed, and a set including 23 benchmark functions in the literature as test beds are employed for this comparison. These test functions consist of 3-group, namely unimodal (F1–F7), multi-modal (F8–F13), and fixed-dimension (F14–F23) multimodal benchmark functions. POA is compared with seven algorithms, namely PSO1, GWO5, GSA27, FA2 and ASO28, HHO11, HSG29 on a set of 23 benchmark functions as shown in Table 1.

Numerical examples with Dim ≤ 30. Each benchmark function runs 30 times by the POA algorithm. A sample size of POA with 30 planets is selected to perform 500 iterations. The statistical results (average–Ave, and standard deviation–Std) are summarized in Tables 2, 3 and 4.

The results from the comparison to F5 and F6 are quite good for POA compared with the others. The F1 to F4 and F7 functions witness that POA algorithm's accuracy is the most superior compared with all other algorithms.

In comparison with 7-unimodal functions, the most of multimodal functions consist of a lot of local optimization areas with the number increasing exponentially with dimensions. This makes them good conditions to evaluate the exploratory ability of a meta-heuristic optimization algorithm.

Table 3 indicates that POA outperforms in F10 and F11 functions, and is quite competitive with the rest.

Similar to the unimodal functions, once again the multimodal and fixed-dimension multimodal functions prove the competitiveness of POA with other algorithms, and show that the obtained results from F14 to F23 are promising.

Figure 6 illustrates the convergence of POA after 100 iterations of 100 planets. The first two metrics are qualitative metrics that illustrate the history of planets through the course of generations. During the whole optimization process, the planets are represented using red points as shown in Fig. 6. The trend of planets explores potential zones of the search space, and exploit quite accurately the global optimum. These investigations demonstrate that POA is able to get the high effectiveness in approximating the global optimum of optimization problems.

The third metric presents the movement of the 1st planet in the first dimension during optimization. This metric helps us to monitor if the first planet, which represents all planets, faces sudden movement in the initial generations and has more stability in the final generations. This movement is able to guarantee the exploration of the search region. Finally, the movement of planets is very short, which causes exploitation of the search region.

Obviously, POA demonstrates that this is an algorithm that meets a requirement of accuracy, as well as a high degree of convergence.

The final quantitative metric is the convergence level of the POA algorithm. The best value of all planets in each generation is stored and the convergence curves are shown in Fig. 6. The decreasing of fitness over the generations demonstrates the convergence of the POA algorithm.

Numerical examples with high-dimensional optimization problems. To validate the performance of POA with respect to high-dimensional optimization problems, the first 13 classical benchmark functions of the above-mentioned ones with Dim = 1000 are employed to investigate POA. For a fair comparison, seven of the above mentioned meta-heuristic optimization algorithms and POA with population size $N = 30$ independently run in 30 times. Additionally, the maximum number of iterations is fixed at 500 for all test functions.
Table 1. Classical benchmark functions.

| Function ($F_i$)                                                                 | Range          | Dim | Min$_x$ |
|--------------------------------------------------------------------------------|----------------|-----|---------|
| $F_1 = \sum_{i=1}^n (x_i)^2$                                                   | $[-100,100]$   | 30  | 0       |
| $F_2 = \sum_{i=1}^n |x_i| + \prod_{i=1}^n |x_i|$                                                    | $[-10,10]$     | 30  | 0       |
| $F_3 = \sum_{i=1}^n \left( \sum_{j=1}^i x_j \right)^2$                       | $[-100,100]$   | 30  | 0       |
| $F_4 = \max_i \{|x_i|, 1 \leq i \leq n\}$                                     | $[-100,100]$   | 30  | 0       |
| $F_5 = \sum_{i=1}^{n-1} \left( 100(x_{i+1} - x_i)^2 + (x_i - 1)^2 \right)$ | $[-30,30]$    | 30  | 0       |
| $F_6 = \sum_{i=1}^n (x_i + 0.5)^2$                                             | $[-100,100]$   | 30  | 0       |
| $F_7 = \sum_{i=1}^n x_i + \text{rand}(0, 1)$                                  | $[-1.28,1.28]$ | 30  | 0       |
| $F_8 = \sum_{i=1}^n -x_i \sin \sqrt{|x_i|}$                                   | $[-500,500]$   | 30  | $-418.9829 \times \text{Dim}$ |
| $F_9 = \sum_{i=1}^n (x_i^2 - 10 \cos(2\pi x_i) + 10)^2$                      | $[-5.12,5.12]$ | 30  | 0       |
| $F_{10} = -20 \exp \left(-0.2 \sqrt{\sum_{i=1}^n x_i^2} - \exp \left( \frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i) \right) + 20 + c \right)$ | $[-32,32]$    | 30  | 0       |
| $F_{11} = \frac{1}{\sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos \left( \frac{x_i}{n} \right) + 1}$ | $[-600,600]$  | 30  | 0       |
| $F_{12} = \frac{n}{\pi} \left\{ 10 \sin^2(\pi y_1) + \sum_{i=1}^{n-1} \left[ (y_i - 1)^2 \left[ 1 + 10 \sin^2(\pi y_{i+1}) \right] + (y_i - 1)^2 \right] + \sum_{i=1}^n w(x_i, 10, 100, 4) \right\}$ | $[-50,50]$    | 30  | 0       |
| where $y_i = 1 + \frac{(x_i + 1)}{4}$                                        |                |     |         |
| $F_{13} = 0.1 \sin^2(3\pi x_1) + \sum_{i=1}^n (x_i - 1)^2 \left[ 1 + 10 \sin^2(3\pi x_i + 1) \right] + (x_n - 1)^2 \left[ 1 + \sin^2(2\pi x_n) \right] + \sum_{i=1}^n w(x_i, 5, 100, 4)$ | $[-50,50]$    | 30  | 0       |
| $F_{14} = \left( \frac{1}{\prod_{i=1}^n (1 + \sum_{j=1}^{i-1} x_j - x_i^2)} \right)^{-1}$ | $[-65,65]$    | 2   | 1       |
| $F_{15} = \sum_{i=1}^{11} a_i \sqrt{x_i^2 + x_i x_{i+1} + \sum_{j=i+1}^{n} x_j^2}$ | $[-5.5]$       | 4   | 0.00030 |
| $F_{16} = 4x_1^2 - 2.1x_1^4 + \frac{\sum_{i=1}^n x_i^2}{\sum_{i=1}^n x_i^2}$ | $[-5.5]$       | 2   | $-1.0316$ |
| $F_{17} = \left( x_2 - \frac{\sum_{i=1}^n x_i^2}{\sum_{i=1}^n x_i} - 6 \right) + 10 \left( 1 - \frac{\sum_{i=1}^n x_i^2}{\sum_{i=1}^n x_i} \right) \cos x_3 + 10$ | $[-5.5]$       | 2   | 0.398   |
| $F_{18} = \left[ 1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 16x_2 + 3x_2^2) \right] \times \left[ 30 + (2x_1 - 3x_2)^2 \times \left( 18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2 \right) \right]$ | $[-5.5]$       | 2   | 3       |
| $F_{19} = -\sum_{i=1}^{n} c_i \exp \left( -\sum_{j=1}^{n} a_j (x_j - p_j)^2 \right)$ | $[1.3]$        | 3   | $-3.86$ |
| $F_{20} = -\sum_{i=1}^{n} c_i \exp \left( -\sum_{j=1}^{n} a_j (x_j - p_j)^2 \right)$ | $[0.1]$        | 6   | $-3.32$ |
| $F_{21} = \sum_{i=1}^{n} (X - A_i) (X - a_i)^2 + c_i$ | $[0.10]$       | 4   | $-10.1532$ |
| $F_{22} = \sum_{i=1}^{n} (X - a_i) (X - a_i)^2 + c_i$ | $[0.10]$       | 4   | 10.4028 |
| $F_{23} = \sum_{i=1}^{n} (X - a_i) (X - a_i)^2 + c_i$ | $[0.10]$       | 4   | 10.5363 |

The test discussed in this subsection demonstrate that POA is promising for dealing with 13 classical benchmark problems. Among the tested 23 benchmark functions, 13 functions had Dim $= 1000$, as presented in Table 5 and Fig. 7. This subsection confirmed the ability of POA to deal with high-dimensional problems as the dimension of those 13 classical benchmarks has been increased from 30 to 1000.

Wall-clock time analysis. In this experiment, a comparison is made between POA and the other seven algorithms in the time-consuming computation experiments of the 13 functions. The time-consuming calculation
method is that each benchmark function independently implements 30-times all algorithms, then the values of 30-time running is saved in Table 6. For Dim = 30, not only does the computation of POA outperform some algorithms, while taking less time, such as GSA, ASO, and FA, but also it is sometimes far superior to GWO, even the time-consuming calculation of PSO. For Dim = 1000, POA always ranks first in computational time. These results show that the POA has merit for optimization problems in high dimensional problems.

**Experimental results using CEC functions.** In order to further clarify the efficiency of the proposed algorithm, POA is tested on the complex challenges, namely Evaluation Criteria for the CEC 2017 and CEC 2019. Its results are compared with those of well-known and modern meta-heuristic algorithms: DA, WOA, and the arithmetic optimization algorithm (AOA). These algorithms are selected because of the reasons:

- All of them are based on the principle of PSO as with POA.
- All algorithms are well cited in the literature, and AOA is a recently published study.
- These algorithms were proven that they were superior performance both on benchmark test functions and real-world problems.
- They are publicly provided by their authors.

Like the 23 classical benchmark functions, each function of the CEC Benchmark Suite is run 30 times, and each algorithm was allowed to search the landscape for 500 iterations using 30 agents.

**CEC 2017 problems.** In this subsection, the IEEE CEC 2017 problems is employed to test the performance of POA. The CEC’17 standard set consists of 28 real challenging benchmark problems. The first is unimodal function, 2–7 are multimodal one. While ten functions next are Hybrid, the rest of CEC 2017 are 10 composition functions. All algorithms are well cited in the literature, and AOA is a recently published study.

Table 6 presents a brief description of CEC 2017. In this subsection, the IEEE CEC 2017 problems is employed to test the performance of POA. The CEC’17 standard set consists of 28 real challenging benchmark problems. The first is unimodal function, 2–7 are multimodal one. While ten functions next are Hybrid, the rest of CEC 2017 are 10 composition functions. All of them are based on the principle of PSO as with POA.

**CEC 2019 problems.** Table 10 presents a brief description of CEC 2019. It can be seen from Table 11 that POA outperforms other optimization algorithms in all CEC 2019 functions. Indeed, results in many test functions (e.g. F52, F53, F56) show that POA is more powerful than others not only at the average value of 30 runs, but also at the other statistical values, such as the best, worst and Std value. Once again, The Wilcoxon signed rank test (as shown in Table 12) demonstrated the superior performance of POA to solve CEC 2019 problems.

In the next section, some classical engineering design problems are employed to further evaluate the performance of the POA. Besides, POA is also compared with other well-known techniques to confirm its results.
Table 4. Results of fixed-dimension multimodal benchmark functions. Significant values are in bold.
| $F_i$ | 3D view for two-dimensional version | Population at the start | Population at the end | Trajectory 1$^{st}$ dimension | Convergence curve |
|------|-----------------------------------|------------------------|-----------------------|-------------------------------|------------------|
| $F1$ | ![3D view](image1.png)           | ![Population start](image2.png) | ![Population end](image3.png) | ![Trajectory](image4.png) | ![Convergence](image5.png) |
| $F7$ | ![3D view](image6.png)           | ![Population start](image7.png) | ![Population end](image8.png) | ![Trajectory](image9.png) | ![Convergence](image10.png) |
| $F8$ | ![3D view](image11.png)          | ![Population start](image12.png) | ![Population end](image13.png) | ![Trajectory](image14.png) | ![Convergence](image15.png) |
| $F9$ | ![3D view](image16.png)          | ![Population start](image17.png) | ![Population end](image18.png) | ![Trajectory](image19.png) | ![Convergence](image20.png) |
| $F10$| ![3D view](image21.png)          | ![Population start](image22.png) | ![Population end](image23.png) | ![Trajectory](image24.png) | ![Convergence](image25.png) |
| $F11$| ![3D view](image26.png)          | ![Population start](image27.png) | ![Population end](image28.png) | ![Trajectory](image29.png) | ![Convergence](image30.png) |
| $F14$| ![3D view](image31.png)          | ![Population start](image32.png) | ![Population end](image33.png) | ![Trajectory](image34.png) | ![Convergence](image35.png) |
| $F15$| ![3D view](image36.png)          | ![Population start](image37.png) | ![Population end](image38.png) | ![Trajectory](image39.png) | ![Convergence](image40.png) |
| $F18$| ![3D view](image41.png)          | ![Population start](image42.png) | ![Population end](image43.png) | ![Trajectory](image44.png) | ![Convergence](image45.png) |

**Figure 6.** Level of convergence of POA after 100 iterations.
In this study, three constrained engineering design problems, namely tension/compression spring, welded beam, pressure vessel designs, are used to investigate the applicability of POA. The problems have some equality and inequality constraints. The POA should, therefore, be equipped with a constraint solving technique. Meanwhile, POA can optimize constrained problems as well at the same time. It should be noted that the population size and the number of iterations are, respectively, set to 30 and 50 runs to find the results for all problems in this section.

**Tension/compression spring.** The main aim of this problem is to minimize the weight of a tension/compression spring. The design problem is subject to three constraints, namely surge frequency, shear stress, and minimum deflection. This problem consists of three variables: Wire diameter \(d\), mean coil diameter \(D\), and the number of active coils \(N\).

Table 5. Results of the first 13 benchmark functions with Dim = 1000. Significant values are in bold.

| Fi  | POA | PSO | GSA | GWO | ASO | FA | HHO | HGS |
|-----|-----|-----|-----|-----|-----|----|-----|-----|
|     | Aver | Std | Aver | Std | Aver | Std | Aver | Std |
| F1  | 1.84E−235 | 0 | 2,907,289 | 144,955 | 126,322.75 | 126.32275 | 329,872.3 | 27,485.71 |
| F2  | 9.66E−130 | 5.3E−129 | 2.4E+109 | 1.3E+110 | 806,94152 | 64,64325 | 4.94E−07 | 1.91E−07 |
| F3  | 7.19E−36  | 3.94E−35 | 25,298,720 | 3,619,734 | 26,256,897 | 15,611,478 | 1,387,902 | 256,574.3 |
| F4  | 1.07E−106 | 4.5E−106 | 99,60164 | 0.126,646 | 35,202087 | 1,600084 | 79,1974 | 3.45111 |
| F5  | 998,90393 | 0.060344 | 2,89E+13 | 5.65E+11 | 3,102E+12 | 806,94152 | 399,8334 | 1851,481 |
| F6  | 207,83991 | 6.87E+066 | 2,867,223 | 179,732.3 | 125,969,19 | 5451,924 | 203,1294 | 2,189,645 |
| F7  | 0.090175 | 0.000123 | 240,456.4 | 5599,434 | 5621,5092 | 684,4318 | 140,4884 | 0.029257 |
| F8  | 106,369.8 | 7677,324 | 96,426,4 | 4614,017 | 130,880,56 | 882,5566 | 85,658.3 | 18,724.48 |
| F9  | 0 | 0 | 15,455,68 | 765,3185 | 6588,5853 | 198,471 | 221,053 | 60,1763 |
| F10 | 8.882E−16 | 0 | 20,70236 | 0.423177 | 10,911,026 | 0.180,167 | 12,466E+18 | 0.12408 |
| F11 | 0 | 0 | 26,369,83 | 1546,641 | 21,603,807 | 136,2224 | 0.020575 | 0.028735 |
| F12 | 0.7970388 | 0.036699 | 3.63E+10 | 1.02E+09 | 133,193,57 | 106,387,3 | 1,232,015 | 0.285816 |
| F13 | 99.567897 | 0.180338 | 6.62E+10 | 1.59E+09 | 12,570,826 | 2,297,987 | 119,6103 | 0.616993 |

**Figure 7.** Performance comparison of algorithms.

**Engineering design problems.** In this study, three constrained engineering design problems, namely tension/compression spring, welded beam, pressure vessel designs, are used to investigate the applicability of POA. The problems have some equality and inequality constraints. The POA should be, therefore, equipped with a constraint solving technique. Meanwhile, POA can optimize constrained problems as well at the same time. It should be noted that the population size and the number of iterations are, respectively, set to 30 and 50 for 50 runs to find the results for all problems in this section.

**Tension/compression spring.** The main aim of this problem is to minimize the weight of a tension/compression spring. The design problem is subject to three constraints, namely surge frequency, shear stress, and minimum deflection. This problem consists of three variables: Wire diameter \(d\), mean coil diameter \(D\), and the number of active coils \(N\).

Tension/compression spring design problem has been solved by both mathematicians and heuristic techniques. Some researchers have made efforts to employ several methods for minimizing the weight of a tension/compression spring (Ha and Wang: PSO33; Coello and Montes: The Evolution Strategy (ES)34 and GA35; Mahdavi et al.: Harmony Search (HS)36; Belegundu: Mathematical optimization37 and Arora: Constraint correction38;
Huang et al.: Differential Evolution (DE)\textsuperscript{39}). Additionally, GWO\textsuperscript{5} algorithms and HHO\textsuperscript{11} have also been employed as heuristic optimizers for this problem. The comparison of the results of these methods and POA is shown in Table 13.

### Table 6. Wall-clock time costs (second) on benchmarks of POA and other participants.

| Fi | Dim = 30 | Dim = 1000 |
|----|----------|------------|
| FA | GSA | ASO | PSO | GWO | HHO | HGS | POA | FA | GSA | ASO | PSO | GWO | HHO | HGS | POA |
| F1 | 6.52 | 7.51 | 11.45 | 1.47 | 2.71 | 1.69 | 1.78 | 1.74 | 72.65 | 150.28 | 91.24 | 11.16 | 51.21 | 10.77 | 35.28 | 8.21 |
| F2 | 6.31 | 7.56 | 11.46 | 1.52 | 2.84 | 1.56 | 1.88 | 1.79 | 52.92 | 1703.72 | 196.63 | 15.25 | 105.74 | 12.76 | 30.90 | 8.23 |
| F3 | 8.44 | 9.59 | 13.65 | 3.79 | 5.09 | 8.40 | 4.04 | 4.03 | 785.54 | 933.45 | 756.52 | 326.68 | 364.90 | 755.73 | 344.42 | 326.83 |
| F4 | 6.07 | 7.31 | 11.10 | 1.41 | 2.69 | 1.95 | 1.69 | 1.68 | 126.96 | 311.10 | 196.63 | 15.25 | 105.74 | 12.76 | 30.90 | 8.23 |
| F5 | 6.38 | 7.59 | 11.45 | 1.71 | 2.97 | 3.09 | 2.01 | 1.94 | 104.48 | 328.69 | 201.28 | 22.47 | 108.81 | 18.67 | 32.05 | 12.88 |
| F6 | 6.08 | 7.31 | 11.16 | 1.41 | 2.68 | 2.30 | 1.70 | 1.67 | 154.86 | 336.07 | 299.36 | 21.30 | 95.95 | 15.99 | 31.44 | 8.38 |
| F7 | 7.64 | 8.73 | 12.76 | 2.89 | 4.17 | 5.36 | 3.19 | 3.14 | 189.12 | 333.90 | 223.45 | 55.09 | 76.02 | 52.57 | 49.73 | 39.10 |
| F8 | 6.45 | 7.60 | 11.16 | 1.76 | 3.09 | 4.04 | 2.03 | 2.07 | 170.21 | 333.92 | 288.38 | 36.22 | 114.84 | 32.51 | 37.06 | 26.53 |
| F9 | 6.35 | 7.47 | 11.95 | 1.67 | 2.82 | 2.69 | 3.60 | 1.69 | 78.21 | 154.78 | 95.58 | 17.93 | 53.37 | 24.50 | 362.97 | 12.00 |
| F10 | 6.37 | 7.47 | 11.45 | 1.73 | 2.81 | 2.81 | 3.21 | 1.76 | 78.86 | 158.87 | 95.81 | 17.60 | 52.84 | 25.08 | 361.28 | 13.66 |
| F11 | 7.21 | 8.25 | 14.26 | 1.96 | 3.04 | 3.25 | 3.74 | 1.99 | 102.20 | 157.92 | 169.01 | 20.61 | 55.02 | 30.96 | 417.91 | 14.60 |
| F12 | 10.21 | 11.25 | 15.33 | 5.55 | 6.70 | 12.24 | 5.75 | 5.82 | 114.12 | 188.42 | 131.18 | 53.23 | 90.17 | 113.10 | 152.32 | 48.53 |
| F13 | 10.35 | 11.33 | 15.38 | 5.60 | 6.70 | 12.28 | 5.75 | 5.79 | 114.48 | 189.23 | 131.45 | 53.13 | 89.94 | 115.27 | 171.67 | 43.90 |
| Sum | 94.39 | 108.95 | 162.88 | 32.47 | 48.30 | 61.65 | 40.37 | 35.11 | 2144.60 | 5280.36 | 6302.91 | 662.26 | 1310.59 | 1219.29 | 2059.67 | 571.47 |
| Ranking | 6 | 7 | 8 | 1 | 4 | 5 | 3 | 2 | 6 | 7 | 8 | 2 | 4 | 3 | 5 | 1 |

### Table 7. CEC 2017 problems.

| Type function | Fi | Function name | Range | Dim | Min$_x$
|---------------|----|---------------|-------|-----|---------|
| Unimodal functions | F24 | Shifted and Rotated Bent Cigar Function | | 100 | 100 |
| | F25 | Shifted and Rotated Rosenbrock’s Function | | 300 | |
| | F26 | Shifted and Rotated Rastrigin’s Function | | 400 | |
| Simple multimodal functions | F27 | Shifted and Rotated Expanded Scaffer’s F7 Function | [-100,100] | 10 | 500 |
| | F28 | Shifted and Rotated Lunacek Bi_Rastrigin Function | | 600 | |
| | F29 | Shifted and Rotated Non-Continuous Rastrigin’s Function | | 700 | |
| Hybrid functions | F30 | Shifted and Rotated Levy Function | | 800 | |
| | F31 | Shifted and Rotated Schwefel’s Function | | 900 | |
| Composition functions | F32 | Hybrid Function 1 ($N=3$) | | 1000 | |
| | F33 | Hybrid Function 2 ($N=3$) | | 1100 | |
| | F34 | Hybrid Function 3 ($N=3$) | | 1200 | |
| | F35 | Hybrid Function 4 ($N=4$) | | 1300 | |
| | F36 | Hybrid Function 5 ($N=4$) | | 1400 | |
| | F37 | Hybrid Function 6 ($N=4$) | | 1500 | |
| | F38 | Hybrid Function 7 ($N=5$) | | 1600 | |
| | F39 | Hybrid Function 8 ($N=5$) | | 1700 | |
| | F40 | Hybrid Function 9 ($N=5$) | | 1800 | |
| | F41 | Hybrid Function 10 ($N=6$) | | 1900 | |
| Composition functions | F42 | Composition Function 1 ($N=3$) | | 2000 | |
| | F43 | Composition Function 2 ($N=3$) | | 2100 | |
| | F44 | Composition Function 3 ($N=3$) | | 2200 | |
| | F45 | Composition Function 4 ($N=4$) | | 2300 | |
| | F46 | Composition Function 5 ($N=5$) | | 2400 | |
| | F47 | Composition Function 6 ($N=5$) | | 2500 | |
| | F48 | Composition Function 7 ($N=6$) | | 2600 | |
| | F49 | Composition Function 8 ($N=6$) | | 2700 | |
| | F50 | Composition Function 9 ($N=3$) | | 2800 | |
| | F51 | Composition Function 10 ($N=3$) | | 2900 | |

Huang et al.: Differential Evolution (DE)\textsuperscript{39}). Additionally, GWO\textsuperscript{5} algorithms and HHO\textsuperscript{11} have also been employed as heuristic optimizers for this problem. The comparison of the results of these methods and POA is shown in Table 13.
Table 8. Results of CEC 2017 problems. Significant values are in bold.
**Welded beam.** The aim of welded beam design problem is to minimize its fabrication cost. The constraints of the problem are shear stress ($\tau$), bending stress in the beam ($\theta$), buckling load of the bar ($P_c$), end deflection of the beam ($\delta$) and side constraints. Welded beam design problem has four variables, namely thickness of weld ($h$), length of attached part of bar ($l$), the height of the bar ($t$), and thickness of the bar ($b$). This problem is illustrated in the literature\(^5\,\text{40,41}\).

Lee and Geem\(^40\) employed HS to deal with this problem, while Deb\(^42,43\) and Coello\(^44\) used GA. Seyedali Mirjalili applied GWO\(^5\) to solve this problem. Richardson's random approach, Davidon-Fletcher-Powell, Simplex

| Fi | Function name | Range           | Dim | Min$_{F}$ |
|----|--------------|----------------|-----|-----------|
| F52 | Storn's Chebyshev Polynomial Fitting Problem | $[-8192, 8192]$ | 9   | 1         |
| F53 | Inverse Hilbert Matrix Problem | $[-16384, 16384]$ | 16  | 1         |
| F54 | Lennard–Jones Minimum Energy Cluster | $[-4,4]$ | 18  | 1         |
| F55 | Rastrigin's Function | $[-100,100]$ | 10  | 1         |
| F56 | Griewangk's Function | $[-100,100]$ | 10  | 1         |
| F57 | Weierstrasse Function | $[-100,100]$ | 10  | 1         |
| F58 | Modified Schwefel's Function | $[-100,100]$ | 10  | 1         |
| F59 | Expanded Schaffer's F6 Function | $[-100,100]$ | 10  | 1         |
| F60 | Happy Cat Function | $[-100,100]$ | 10  | 1         |
| F61 | Ackley Function | $[-100,100]$ | 10  | 1         |

**Table 10.** CEC 2019 problems.

---

## Table 9. Results of Wilcoxon sign-rank test for CEC 2017 problems with $\alpha = 0.05$.

| Fi | POA vs WOA $p$-value | Winner | POA vs DA $p$-value | Winner | POA vs AOA $p$-value | Winner |
|----|------------------|--------|------------------|--------|------------------|--------|
| F24 | 3.02E–11 | + | 3.02E–11 | + | 3.02E–11 | + |
| F25 | 3.02E–11 | + | 3.02E–11 | + | 3.02E–11 | + |
| F26 | 2.20E–07 | + | 1.73E–07 | + | 3.34E–11 | + |
| F27 | 2.34E–01 | ~ | 3.50E–03 | + | 3.50E–03 | + |
| F28 | 1.37E–03 | + | 1.99E–02 | + | 2.13E–04 | + |
| F29 | 2.71E–02 | + | 2.80E–05 | ~ | 8.35E–08 | + |
| F30 | 1.12E–01 | ~ | 7.73E–01 | ~ | 4.84E–02 | + |
| F31 | 3.83E–05 | + | 9.94E–01 | ~ | 6.55E–04 | + |
| F32 | 5.30E–01 | ~ | 7.98E–02 | ~ | 7.98E–02 | ~ |
| F33 | 1.32E–04 | + | 2.49E–06 | + | 2.15E–10 | + |
| F34 | 1.61E–06 | + | 1.25E–05 | + | 6.12E–10 | + |
| F35 | 4.22E–04 | + | 1.32E–04 | + | 1.41E–04 | + |
| F36 | 1.70E–02 | ~ | 1.56E–02 | ~ | 5.79E–01 | ~ |
| F37 | 7.04E–07 | + | 5.00E–09 | + | 2.15E–10 | + |
| F38 | 8.42E–01 | ~ | 2.58E–01 | ~ | 1.17E–02 | + |
| F39 | 1.08E–02 | + | 4.22E–04 | + | 3.16E–05 | + |
| F40 | 3.79E–01 | ~ | 7.96E–01 | ~ | 7.06E–01 | ~ |
| F41 | 5.83E–03 | + | 3.48E–01 | ~ | 8.35E–08 | + |
| F42 | 4.38E–01 | ~ | 8.77E–01 | ~ | 1.91E–01 | ~ |
| F43 | 7.73E–01 | ~ | 8.77E–01 | ~ | 5.01E–01 | ~ |
| F44 | 1.78E–04 | + | 4.64E–05 | + | 6.33E–08 | + |
| F45 | 5.83E–03 | + | 8.20E–07 | + | 4.50E–11 | + |
| F46 | 9.33E–02 | ~ | 4.08E–05 | + | 9.51E–06 | + |
| F47 | 1.53E–05 | + | 1.30E–03 | + | 3.02E–11 | + |
| F48 | 5.37E–02 | ~ | 2.52E–01 | ~ | 3.37E–04 | + |
| F49 | 2.12E–01 | ~ | 4.12E–01 | ~ | 1.41E–09 | + |
| F50 | 1.17E–02 | + | 2.84E–01 | ~ | 1.78E–10 | + |
| F51 | 5.55E–02 | ~ | 8.53E–01 | ~ | 6.10E–03 | + |
| Sum (+/~/−) | 16/11/1 | 13/12/3 | 23/5/0 |

---

---
technique, Griffith and Stewart's successive linear approximation are the mathematical methods that have been adopted by Ragsdell and Philips for this problem. More recently, Heidari, et al. and Yang, et al. have used HHO and HGS, respectively, to solve the problem. Table 14 shows a comparison between the different methods. The results indicate that POA reaches a design with the minimum cost compared to other optimizations. The best result of the cost function obtained by the POA is $1.72564$.

Pressure vessel. Pressure vessel design problem is well-known, where the fabrication cost of the total cost consisting of material, forming, and welding of a cylindrical vessel should be minimized. There are four variables, namely thickness of the shell ($T_s$), thickness of the head ($Th$), Inner radius ($R$) and length of the cylindrical section without considering the head ($L$), and four constraints.

Pressure vessel design problem has also been popular among optimization studies in different researches. Several heuristic techniques, namely DE, PSO, GA, ACO, ES, GWO, MFO, HHO, and SMA, that have been adopted for the optimization of this problem. Mathematical approaches employed are augmented Lagrangian Multiplier and branch-and-bound. We can see that POA is again able to search a design with the minimum cost as shown in Table 15.

Conclusions

In the paper, a meta-heuristic algorithm, inspired by the gravitational law of Newton, is proposed. POA's structure in search processes consists of 2 phases that aim for proper balance exploration and exploitation. Several outperform features are shown through the accuracy of 23 classical benchmark functions and 38 IEEE CEC test.
Table 13. Comparison of results for tension/compression spring. Significant values are in bold.

| Candidates      | Optimum variables | Optimum weight |
|-----------------|-------------------|----------------|
|                 | d                 | D              | N              |
| HHO             | 0.05179639        | 0.35930536     | 11.138859      |
| POA             | 0.051767          | 0.358602       | 11.179891      |
| GWO             | 0.051690          | 0.356737       | 11.288850      |
| MFO             | 0.051994          | 0.364109       | 10.868422      |
| DE (Huang et al.) | 0.051609    | 0.354714       | 11.410831      |
| HS (Mahdavie et al.) | 0.051154 | 0.349871       | 12.076432      |
| PSO (Ha and Wang) | 0.051728     | 0.357644       | 11.244543      |
| ES (Coello and Montes) | 0.051989  | 0.363965       | 10.890522      |
| GSA             | 0.050276          | 0.323680       | 13.525410      |
| GA (Coello)     | 0.051480          | 0.351661       | 11.632701      |
| Mathematical optimization (Belegundu) | 0.051989  | 0.352363       | 11.739891      |
| Constraint correction (Arora) | 0.050000 | 0.315900       | 14.250000      |

Table 14. Comparison of results for welded beam design problem. Significant values are in bold.

| Candidates      | Optimum variables | Optimum cost |
|-----------------|-------------------|--------------|
|                 | (h)               | (l)          |
| POA             | 0.20563           | 3.47242      |
| GWO             | 0.20568           | 3.47383      |
| HHO             | 0.204039          | 3.531061     |
| GA (Coello)     | N.A               | N.A          |
| GSA             | 0.182129          | 3.594799     |
| HGs             | 0.26              | 5.1025       |
| GA (Deb)        | N.A               | N.A          |
| HS (Lee and Geem) | 0.2442       | 6.2231       |
| APPROX          | 0.2444            | 6.2189       |
| David           | 0.2434            | 6.2552       |
| GA (Deb)        | 0.2489            | 6.173        |
| Simplex         | 0.2792            | 5.6256       |
| Random          | 0.4575            | 4.7313       |

Table 15. Comparison of results for pressure vessel design problem.
functions (CEC2017, CEC 2019). In many functions, POA showed that the obtained results are more accurate than the others many times.

In the final evaluation section, a set of well-known test cases, including three engineering test problems, are thoroughly investigated to examine the operation of POA in practice. Each problem is a type of distinct engineering, including very diverse search spaces. Therefore, these engineering problems are employed to test the POA thoroughly. The obtained results demonstrate that POA is able to solve effectively real challenging problems with unknown search spaces and a large number of constraints. The results compared to GSA, GWO, PSO, DE, ACO, MFO, SOS, CS, HHMO, SMA, HGs, etc., suggest that POA is superior.

The structure of POA is simple and explicit, very effective, even fast. Experiments revealed short computational time for handling complex optimization problems. Therefore, we firmly authenticate that POA is a powerful algorithm to solve optimization problems.

Data availability
All data generated or analyzed during this study are included in this published article.

Received: 23 March 2022; Accepted: 4 May 2022
Published online: 19 May 2022

References
1. Kennedy, J. & Eberhart, R. In Proceedings of ICNN’95 - International Conference on Neural Networks. 1942–1948 vol.1944.
2. Yang, X. -S. In Stochastic Algorithms: Foundations and Applications. (eds Osamu Watanabe & Thomas Zeugmann) 169–178 (Springer, Berlin) (2009).
3. Mirjalili, S. Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput. Appl. 27, 1053–1073 (2016).
4. Mirjalili, S. & Lewis, A. The whale optimization algorithm. Adv. Eng. Softw. 95, 51–67 (2016).
5. Mirjalili, S., Mirjalili, S. M. & Lewis, A. Grey wolf optimizer. Adv. Eng. Softw. 69, 46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007 (2014).
6. Wang, G.-G., Deb, S. & Cui, Z. Monarch butterfly optimization. Neural Comput. Appl. 31, 1995–2014 (2019).
7. Wang, G.-G., Deb, S. & Coelho, L. D. S. Earthworm optimisation algorithm: a bio-inspired metaheuristics algorithm for global optimisation problems. Int. J. Bio-Inspired Comput. 12, 1–22 (2018).
8. Wang, G.-G., Deb, S. & Coelho, L. D. S. In 2015 3rd International Symposium on Computational and Business Intelligence (ISCBI). 1–5 (IEEE).
9. Wang, G.-G. Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems. Memetic Comput. 10, 151–164 (2018).
10. Li, S., Chen, H., Wang, M., Heidari, A. A. & Mirjalili, S. Slime mould algorithm: a new method for stochastic optimization. Futur. Gener. Comput. Syst. 111, 300–323 (2020).
11. Heidari, A. A. et al. Harris hawks optimization: algorithm and applications. Futur. Gener. Comput. Syst. 97, 849–872 (2019).
12. Moghaddam, F. F., Moghaddam, R. F. & Cheriet, M. Curved space optimization: a random search based on general relativity theory. arXiv preprint arXiv:1208.2214 (2012).
13. Zheng, Y.-J. Water wave optimization: a new nature-inspired metaheuristic. Comput. Oper. Res. 55, 1–11 (2015).
14. Ahmadifar, I., Heidari, A. A., Gandomi, A. H., Chu, X. & Chen, H. RUN beyond the metaphor: an efficient optimization algorithm based on Runge Kutta method. Expert Syst. Appl. 181, 115079 (2021).
15. Rao, R. V., Savsani, V. J. & Vakarh, D. Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput. Aided Des. 43, 303–315 (2011).
16. Ahmadi, S.-A. Human behavior-based optimization: a novel metaheuristic approach to solve complex optimization problems. Neural Comput. Appl. 28, 233–244 (2017).
17. Goldberg, D. E. Genetic algorithms in search. Optimization, and Machine Learning (1989).
18. Juste, K., Kita, H., Tanaka, E. & Hasegawa, J. An evolutionary programming solution to the unit commitment problem. IEEE Trans. Power Syst. 14, 1452–1459 (1999).
19. Holland, J. H. Outline for a logical theory of adaptive systems. J. ACM 9, 297–314 (1962).
20. Patro, S. P., Nayak, G. S. & Padhy, N. Heart disease prediction by using novel optimization algorithm: a supervised learning perspective, and towards performance shifts. Expert Syst. Appl. 114864 (2021).
21. Li, X. & Sun, Y. Stock intelligent investment strategy based on support vector machine parameter optimization algorithm. Neural Comput. Appl. 32, 1765–1775 (2020).
22. Sang-To, T. et al. Combination of intermittent search strategy and an improve particle swarm optimization algorithm (IPSO) for model updating. Frattura ed Integrità Strutturale 59, 141–152 (2022).
23. Minn, H.-L. et al. In Proceedings of the 2nd International Conference on Structural Damage Modelling and Assessment. 13–26 (Springer).
24. Yao, X., Liu, Y. & Lin, G. Evolutionary programming made faster. IEEE Trans. Evol. Comput. 3, 82–102, doi:https://doi.org/10.1109/4235.771163 (1999).
25. Digalakis, J. G. & Margaritis, K. G. On benchmarking functions for genetic algorithms. Int. J. Comput. Math. 77, 481–506 (2001).
26. Wang, X.-S. Test problems in optimization. arXiv preprint arXiv:1008.0549 (2010).
27. Rashedi, E., Nezamabadi-Pour, H. & Saryazdi, S. GSA: a gravitational search algorithm. Inf. Sci. 179, 2232–2248 (2009).
28. Zhao, W., Wang, L. & Zhang, Z. Atom search optimization and its application to solve a hydrogeologic parameter estimation problem. Knowl.-Based Syst. 163, 283–304. https://doi.org/10.1016/j.knosys.2018.08.030 (2019).
29. Yang, Y., Chen, H., Heidari, A. A. & Gandomi, A. H. Hunger games search: Visions, conception, implementation, deep analysis, and perspectives, and towards performance shifts. Expert Syst. Appl. 177, 114864 (2021).
30. Wu, G., R. Mallipeddi, and P. N. Suganthan. Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization. (researchgate, 2017).
31. Price, K., Awdad, N., Ali, M. & Suganthan, P. The 100-digit challenge: problem definitions and evaluation criteria for the 100-digit challenge special session and competition on single objective numerical optimization. Nanyang Technol. Univ. (2018).
32. Abualigah, L., Diabat, A., Mirjalili, S., Abd Elaziz, M. & Gandomi, A. H. The arithmetic optimization algorithm. Comput. Methods Appl. Mech. Eng. 376, 113609 (2021).
33. He, Q. & Wang, L. An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Eng. Appl. Artif. Intell. 20, 89–99 (2007).
34. Mezura-Montes, E. & Coello, C. A. C. An empirical study about the usefulness of evolution strategies to solve constrained optimization problems. Int. J. Gen Syst 37, 443–473 (2008).
35. Coello, C. A. C. Use of a self-adaptive penalty approach for engineering optimization problems. *Comput. Ind.* **41**, 113–127 (2000).
36. Mahdavi, M., Fesanghary, M. & Damangir, E. An improved harmony search algorithm for solving optimization problems. *Appl. Math. Comput.* **188**, 1567–1579 (2007).
37. Belegundu, A. D. & Arora, J. S. A study of mathematical programming methods for structural optimization. Part I: theory. *Int. J. Numer. Methods Eng.* **21**, 1583–1599, doi:https://doi.org/10.1002/nme.1620210904 (1985).
38. Arora, J. Introduction to optimum design with MATLAB. *Introduction to Optimum Design*, 413–432 (2004).
39. Huang, F.-Z., Wang, L. & He, Q. An effective co-evolutionary differential evolution for constrained optimization. *Appl. Math. Comput.* **186**, 340–356 (2007).
40. Lee, K. S. & Geem, Z. W. A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. *Comput. Methods Appl. Mech. Eng.* **194**, 3902–3933 (2005).
41. Ragsdell, K. M. & Phillips, D. T. Optimal design of a class of welded structures using geometric programming. *J. Eng. Ind.* **98**, 1021–1025. https://doi.org/10.1115/1.3439995 (1976).
42. Deb, K. Optimal design of a welded beam via genetic algorithms. *AIAM J.* **29**, 2013–2015 (1991).
43. Deb, K. An efficient constraint handling method for genetic algorithms. *Comput. Methods Appl Mech Eng* **186**, 311–338 (2000).
44. Coello Coello, C. A. Constraint-handling using an evolutionary multiobjective optimization technique. *Comput. Methods Appl Mech Eng* (2010).
45. Coello, C. A. C. & Montes, E. M. Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Adv. Eng. Inform.* **16**, 193–203 (2002).
46. Deb, K. In *Evolutionary Algorithms in Engineering Applications* 497–514 (Springer, 1997).
47. Kaveh, A. & Talatahari, S. An improved ant colony optimization for constrained engineering design problems. *Eng. Comput.* (2010).
48. Mirjalili, S. Moth–flame optimization algorithm: a novel nature-inspired heuristic paradigm. *Knowl.-Based Syst.* **89**, 228–249 (2015).
49. Kannan, B. K. & Kramer, S. N. An augmented lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. *J. Mech. Des.* **116**, 405–411. https://doi.org/10.1115/1.2919393 (1994).
50. Sandgren, E. Nonlinear integer and discrete optimization in mechanical design optimization. *J. Mech. Des.* **112**, 223–229. https://doi.org/10.1115/1.2912596 (1990).

**Acknowledgements**

The authors acknowledge the financial support of VLIR-UOS TEAM Project, VN2018TEA479A103, ‘Damage assessment tools for Structural Health Monitoring of Vietnamese infrastructures’, funded by the Flemish Government. The authors would like to acknowledge the support from Ho Chi Minh City Open University under the basic research fund (No. E2021.06.1). The authors wish to express their gratitude to Van Lang University, Vietnam for financial support for this research.

**Author contributions**

T.S.T., M.A.W. and T.C.L. Conceptualization, Software, Methodology, Validation, Writing – Original draft; M.L. Writing, formal analysis and data curation; M.A.W. and T.C.L. Supervision.

**Competing interests**

The authors declare no competing interests.

**Additional information**

**Correspondence** and requests for materials should be addressed to M.A.W. or T.C.-L.

**Reprints and permissions information** is available at www.nature.com/reprints.

**Publisher’s note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

---

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

© The Author(s) 2022