S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey Wolf Optimizer, *Advances in Engineering Software* , vol. 69, pp. 46-61, 2014, DOI: http://dx.doi.org/10.1016/j.advengsoft.2013.12.007

Grey Wolf Optimizer

[1] Seyedali Mirjalili, [2] Seyed Mohammad Mirjalili, [1] Andrew Lewis

[1] School of Information and Communication Technology, Griffith University, Nathan, Brisbane, QLD 4111, Australia

[2] Department of Electrical Engineering, Faculty of Electrical and Computer Engineering, Shahid Beheshti University, G. C. 1983963113, Tehran, Iran

seyedali.mirjalili@griffithuni.edu.au, mohammad.smm@gmail.com, a.lewis@griffith.edu.au

***Abstract*:** This work proposes a new meta-heuristic called Grey Wolf Optimizer (GWO) inspired by grey wolves (Canis lupus). The GWO algorithm mimics the leadership hierarchy and hunting mechanism of grey wolves in nature. Four types of grey wolves such as alpha, beta, delta, and omega are employed for simulating the leadership hierarchy. In addition, the three main steps of hunting, searching for prey, encircling prey, and attacking prey, are implemented. The algorithm is then benchmarked on 29 well-known test functions, and the results are verified by a comparative study with Particle Swarm Optimization (PSO), Gravitational Search Algorithm (GSA), Differential Evolution (DE), Evolutionary Programming (EP), and Evolution Strategy (ES). The results show that the GWO algorithm is able to provide very competitive results compared to these well-known meta-heuristics. The paper also considers solving three classical engineering design problems (tension/compression spring, welded beam, and pressure vessel designs) and presents a real application of the proposed method in the field of optical engineering. The results of the classical engineering design problems and real application prove that the proposed algorithm is applicable to challenging problems with unknown search spaces.

Keywords: Optimization, optimization techniques, heuristic algorithm, Metaheuristics, Constrained Optimization, GWO

1. Introduction

Meta-heuristic optimization techniques have become very popular over the last two decades. Surprisingly, some of them such as Genetic Algorithm (GA) [1], Ant Colony Optimization (ACO) [2], and Particle Swarm Optimization (PSO) [3] are fairly well-known among not only computer scientists but also scientists from different fields. In addition to the huge number of theoretical works, such optimization techniques have been applied in various fields of study. There is a question here as to why meta-heuristics have become remarkably common. The answer to this question can be summarized into four main reasons: simplicity, flexibility, derivation-free mechanism, and local optima avoidance.

First, meta-heuristics are fairly simple. They have been mostly inspired by very simple concepts. The inspirations are typically related to physical phenomena, animals' behaviors, or evolutionary concepts. The simplicity allows computer scientists to simulate different natural concepts, propose new meta-heuristics, hybridize two or more meta-heuristics, or improve the current meta-heuristics. Moreover, the simplicity assists other scientists to learn meta-heuristics quickly and apply them to their problems.

Second, flexibility refers to the applicability of meta-heuristics to different problems without any special changes in the structure of the algorithm. Meta-heuristics are readily applicable to different problems since they mostly assume problems as black boxes. In other words, only the input(s) and output(s) of a system are important for a meta-heuristic. So, all a designer needs is to know how to represent his/her problem for meta-heuristics. Encoding the problem is the main issue.

Third, the majority of meta-heuristics have derivation-free mechanisms. In contrast to gradient-based optimization approaches, meta-heuristics optimize problems stochastically. The optimization process starts with random solution(s), and there is no need to calculate the derivative of search spaces to find the optimum. This makes meta-heuristics highly suitable for real problems with expensive or unknown derivative information.

Finally, meta-heuristics have superior abilities to avoid local optima compared to conventional optimization techniques. This is due to the stochastic nature of meta-heuristics which allow them to avoid stagnation in local solutions and search the entire search space extensively. The search space of real problems is usually unknown and very complex with a massive number of local optima, so meta-heuristics are good options for optimizing these challenging real problems.

The No Free Lunch (NFL) theorem [4] is worth mentioning here. This theorem has logically proved that there is no meta-heuristic best suited for solving all optimization problems. In other words, a particular meta-heuristic may show very promising results on a set of problems, but the same algorithm may show poor performance on a different set of problems. Obviously, NFL makes this field of study highly active which results in enhancing current approaches and proposing new meta-heuristics every year. This also motivates our attempts to develop a new meta-heuristic with inspiration from grey wolves.

Generally speaking, meta-heuristics can be divided into two main classes: single-solution-based and population-based. In the former class (Simulated Annealing [5] for instance) the search process starts with one candidate solution. This single candidate solution is then improved over the course of iterations. Population-based meta-heuristics, however, perform the optimization using a set of solutions (population). In this case the search process starts with a random initial population (multiple solutions), and this population is enhanced over the course of iterations. Population-based meta-heuristics have some advantages compared to single solution-based algorithms:

- Multiple candidate solutions share information about the search space which results in sudden jumps toward the promising part of search space
- Multiple candidate solutions assist each other to avoid locally optimal solutions
- Population-based meta-heuristics generally have greater exploration compared to single solution-based algorithms

Crowd intelligence / Swarm Intelligence

One of the interesting branches of the population-based meta-heuristics is Swarm Intelligence (SI). The concepts of SI was first proposed in 1993 [6]. According to Bonabeau *et al.* [1], SI is "*The emergent collective intelligence of groups of simple agents*". The inspirations of SI techniques originate mostly from natural colonies, flock, herds, and schools. Some of the most popular SI techniques are ACO [2], PSO [3], and Artificial Bee Colony (ABC) [7]. A comprehensive literature review of the SI algorithms is provided in the next section. Some of the advantages of SI algorithms are:

- SI algorithms preserve information about the search space over the course of iteration, whereas Evolutionary Algorithms (EA) discard the information of the previous generations
- SI algorithms often utilize memory to save the best solution obtained so far
- SI algorithms usually have fewer parameters to adjust
- SI algorithms have less operators compared to evolutionary approaches (crossover, mutation, elitism, and so on)
- SI algorithms are easy to implement

Regardless of the differences between the meta-heuristics, a common feature is the division of the search process into two phases: [1]exploration and [2]exploitation [8-12]. The exploration phase refers to the process of investigating the promising area(s) of the search space as broadly as possible. An algorithm needs to have stochastic operators to randomly and globally search the search space in order to support this phase. However, exploitation refers to the local search capability around the promising regions obtained in the exploration phase. Finding a proper balance between these two phases is considered a challenging task due to the stochastic nature of meta-heuristics. This work proposes a new SI technique with inspiration from the social hierarchy and hunting behavior of grey wolf packs. The rest of the paper is organized as follows:

Section 2 presents a literature review of SI techniques. Section 3 outlines the proposed GWO algorithm. The results and discussion of benchmark functions, semi-real problems, and a real application are presented in Section 4, Section 5, and Section 6, respectively. Finally, Section 7 concludes the work and suggests some directions for future studies.

2. Literature review

Meta-heuristics may be classified into three main classes: evolutionary, physics-based, and SI algorithms. EAs are usually inspired by the concepts of evolution in nature. The most popular algorithm in this branch is GA. This algorithm was proposed by Holland in 1992 [13] and simulates Darwnian evolution concepts. The engineering applications of GA were extensively investigated by Goldberg [14]. Generally speaking, the optimization is done by evolving an initial random solution in EAs. Each new population is created by the combination and mutation of the individuals in the previous generation. Since the best individuals have higher probability of participating in generating the new population, the new population is likely to be better than the previous generation(s). This can guarantee that the initial random population is optimized over the course of generations. Some of the EAs are Differential Evolution (DE) [15], Evolutionary Programing (EP) [16, 17], and

Evolution Strategy (ES) [18, 19], Probability-Based Incremental Learning (PBIL), Genetic Programming (GP) [20], and Biogeography-Based Optimizer (BBO) [21].

As an example, the BBO algorithm was first proposed by Simon in 2008 [21]. The basic idea of this algorithm has been inspired by biogeography which refers to the study of biological organisms in terms of geographical distribution (over time and space). The case studies might include different islands, lands, or even continents over decades, centuries, or millennia. In this field of study different ecosystems (habitats or territories) are investigated for finding the relations between different species (habitants) in terms of immigration, emigration, and mutation. The evolution of ecosystems (considering different kinds of species such as predator and prey) over migration and mutation to reach a stable situation was the main inspiration of the BBO algorithm.

The second main branch of meta-heuristics is physics-based techniques. Such optimization algorithms typically mimic physical rules. Some of the most popular algorithms are Gravitational Local Search (GLSA) [22], Big-Bang Big-Crunch (BBBC) [23], Gravitational Search Algorithm (GSA) [24], Charged System Search (CSS) [25], Central Force Optimization (CFO) [26], Artificial Chemical Reaction Optimization Algorithm (ACROA) [27], Black Hole (BH) [28] algorithm, Ray Optimization (RO) [29] algorithm, Small-World Optimization Algorithm (SWOA) [30], Galaxy-based Search Algorithm (GbSA) [31], and Curved Space Optimization (CSO) [32]. The mechanism of these algorithms is different from EAs, in that a random set of search agents communicate and move throughout search space according to physical rules. This movement is implemented, for example, using gravitational force, ray casting, electromagnetic force, inertia force, weights, and so on.

For example, the BBBC algorithm was inspired by the big bang and big crunch theories. The search agents of BBBC are scattered from a point in random directions in a search space according to the principles of the big bang theory. They search randomly and then gather in a final point (the best point obtained so far) according to the principles of the big crunch theory. GSA is another physics-based algorithm. The basic physical theory from which GSA is inspired is Newton's law of universal gravitation. The GSA algorithm performs search by employing a collection of agents that have masses proportional to the value of a fitness function. During iteration, the masses are attracted to each other by the gravitational forces between them. The heavier the mass, the bigger the attractive force. Therefore, the heaviest mass, which is possibly close to the global optimum, attracts the other masses in proportion to their distances.

The third subclass of meta-heuristics is the SI methods. These algorithms mostly mimic the social behavior of swarms, herds, flocks, or schools of creatures in nature. The mechanism is almost similar to physics-based algorithm, but the search agents navigate using the simulated collective and social intelligence of creatures. The most popular SI technique is PSO. The PSO algorithm was proposed by Kennedy and Eberhart [3] and inspired from the social behavior of birds flocking. The PSO algorithm employs multiple particles that chase the position of the best particle and their own best positions obtained so far. In other words, a particle is moved considering its own best solution as well as the best solution the swarm has obtained.

Another popular SI algorithm is ACO, proposed by Dorigo et al. in 2006 [2]. This algorithm was inspired by the social behavior of ants in an ant colony. In fact, the social intelligence of ants in finding the shortest path between the nest and a source of food is the main inspiration of ACO. A pheromone matrix is evolved over the course of iteration by the candidate solutions. The ABC is another popular algorithm, mimicking the collective behavior of bees in finding food sources. There are three types of bees in ABS: scout, onlooker, and employed bees. The scout bees are responsible for exploring the search space, whereas onlooker and employed bees exploit the promising solutions found by scout bees. Finally, the Bat-inspired Algorithm (BA), inspired by the echolocation behavior of bats, has been proposed recently [33]. There are many types of bats in the nature. They are different in terms of size and weight, but they all have quite similar behaviors when navigating and hunting. Bats utilize natural sonar in order to do this. The two main characteristics of bats when finding prey have been adopted in designing the BA algorithm. Bats tend to decrease the loudness and increase the rate of emitted ultrasonic sound when they chase prey. This behavior has been mathematically modeled for the BA algorithm. The rest of the SI techniques proposed so far are as follows:

- Marriage in Honey Bees Optimization Algorithm (MBO) in 2001 [34]
- Artificial Fish-Swarm Algorithm (AFSA) in 2003 [35]
- Termite Algorithm in 2005 [36]
- Wasp Swarm Algorithm in 2007 [37]
- Monkey Search in 2007 [38]
- Bee Collecting Pollen Algorithm (BCPA) in 2008 [39]
- Cuckoo Search (CS) in 2009 [40]
- Dolphin Partner Optimization (DPO) in 2009 [41]

- Firefly Algorithm (FA) in 2010 [42]
- Bird Mating Optimizer (BMO) in 2012 [43]
- Krill Herd (KH) in 2012 [44]
- Fruit fly Optimization Algorithm (FOA) in 2012 [45]

This list shows that there are many SI techniques proposed so far, many of them inspired by hunting and search behaviors. To the best of our knowledge, however, there is no SI technique in the literature mimicking the leadership hierarchy of grey wolves, well known for their pack hunting. This motivated our attempt to mathematically model the social behavior of grey wolves, propose a new SI algorithm inspired by grey wolves, and investigate its abilities in solving benchmark and real problems.

3. Grey Wolf Optimizer (GWO)

In this section the inspiration of the proposed method is first discussed. Then, the mathematical model is provided.

3.1. Inspiration

Grey wolf (Canis lupus) belongs to Canidae family. Grey wolves are considered as apex predators, meaning that they are at the top of the food chain. Grey wolves mostly prefer to live in a pack. The group size is 5-12 on average. Of particular interest is that they have a very strict social dominant hierarchy as shown in Fig. 1.

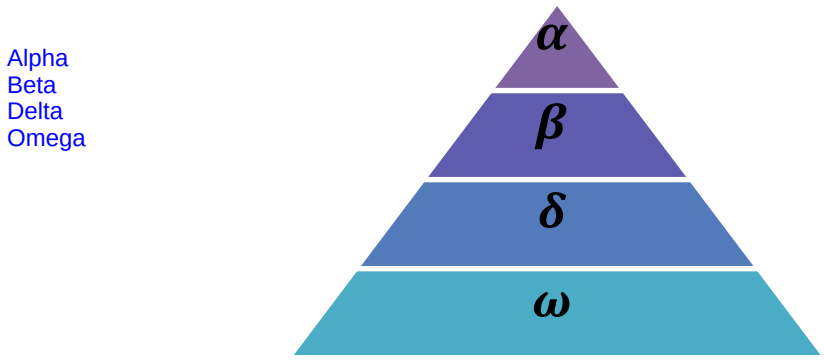


Fig. 1. *Hierarchy of grey wolf (dominance decreases from top down)*

The leaders are a male and a female, called alphas. The alpha is mostly responsible for making decisions about hunting, sleeping place, time to wake, and so on. The alpha's decisions are dictated to the pack. However, some kind of democratic behavior has also been observed, in which an alpha follows the other wolves in the pack. In gatherings, the entire pack acknowledges the alpha by holding their tails down. The alpha wolf is also called the dominant wolf since his/her orders should be followed by the pack [46]. The alpha wolves are only allowed to mate in the pack. Interestingly, the alpha is not necessarily the strongest member of the pack but the best in terms of managing the pack. This shows that the organization and discipline of a pack is much more important than its strength.

The second level in the hierarchy of grey wolves is beta. The betas are subordinate wolves that help the alpha in decision-making or other pack activities. The beta wolf can be either male or female, and he/she is probably the best candidate to be the alpha in case one of the alpha wolves passes away or becomes very old. The beta wolf should respect the alpha, but commands the other lower-level wolves as well. It plays the role of an advisor to the alpha and discipliner for the pack. The beta reinforces the alpha's commands throughout the pack and gives feedback to the alpha.

The lowest ranking grey wolf is omega. The omega plays the role of scapegoat. Omega wolves always have to submit to all the other dominant wolves. They are the last wolves that are allowed to eat. It may seem the omega is not an important individual in the pack, but it has been observed that the whole pack face internal fighting and problems in case of losing the omega. This is due to the venting of violence and frustration of all wolves by the omega(s). This assists satisfying the entire pack and maintaining the dominance structure. In some cases the omega is also the babysitters in the pack.

If a wolf is not an alpha, beta, or omega, he/she is called subordinate (or delta in some references). Delta wolves have to submit to alphas and betas, but they dominate the omega. Scouts, sentinels, elders, hunters, and

caretakers belong to this category. Scouts are responsible for watching the boundaries of the territory and warning the pack in case of any danger. Sentinels protect and guarantee the safety of the pack. Elders are the experienced wolves who used to be alpha or beta. Hunters help the alphas and betas when hunting prey and providing food for the pack. Finally, the caretakers are responsible for caring for the weak, ill, and wounded wolves in the pack.

In addition to the social hierarchy of wolves, group hunting is another interesting social behavior of grey wolves. According to Muro *et al.* [47] the main phases of grey wolf hunting are as follows:

- Tracking, chasing, and approaching the prey
- Pursuing, encircling, and harassing the prey until it stops moving
- Attack towards the prey

These steps are shown in Fig. 2.

Fig. 2. *Hunting behaviour of grey wolves: (A) chasing, approaching, and tracking prey (B-D) pursuiting, harassing, and encircling (E) stationary situation and attack [47]*

In this work this hunting technique and the social hierarchy of grey wolves are mathematically modeled in order to design GWO and perform optimization.

3.2. Mathematical model and algorithm

In this subsection the mathematical models of the social hierarchy, tracking, encircling, and attacking prey are provided. Then the GWO algorithm is outlined.

3.2.1. Social hierarchy:

In order to mathematically model the social hierarchy of wolves when designing GWO, we consider the fittest solution as the alpha (α). Consequently, the second and third best solutions are named beta (β) and delta (δ) respectively. The rest of the candidate solutions are assumed to be omega (ω). In the GWO algorithm the hunting (optimization) is guided by α, β, and δ. The ω wolves follow these three wolves.

3.2.2. Encircling prey:

As mentioned above, grey wolves encircle prey during the hunt. In order to mathematically model encircling behavior the following equations are proposed:

$$\vec{D} = |\vec{C}.\overrightarrow{X_p}(t) - \vec{X}(t)| \quad (3.1)$$

$$\vec{X}(t+1) = \overrightarrow{X_p}(t) - \vec{A}.\vec{D} \quad (3.2)$$

where t indicates the current iteration, $\vec{A}$ and $\vec{C}$ are coefficient vectors , $\overrightarrow{X_p}$ is the position vector of the prey, and $\vec{X}$ indicates the position vector of a grey wolf.

The vectors $\vec{A}$ and $\vec{C}$ are calculated as follows:

$$\vec{A} = 2\vec{a}.\overrightarrow{r_1} - \vec{a} \quad (3.3)$$

We can pass these parts as well

$$\vec{C} = 2.\overrightarrow{r_2} \quad (3.4)$$

where components of $\vec{a}$ are linearly decreased from 2 to 0 over the course of iterations and r_1, r_2 are random vectors in [0,1].

To see the effects of equations (3.1) and (3.2), a two-dimensional position vector and some of the possible neighbors are illustrated in Fig. 3 (a). As can be seen in this figure, a grey wolf in the position of (*X*,*Y*) can update its position according to the position of the prey (*X**,*Y**). Different places around the best agent can be reached with respect to the current position by adjusting the value of $\vec{A}$ and $\vec{C}$ vectors. For instance, (*X*-X*,*Y**) can be reached by setting $\vec{A} = (1,0)$ and $\vec{C} = (1,1)$. The possible updated positions of a grey wolf in 3D space are depicted in Fig. 3 (b). Note that the random vectors r_1 and r_2 allow wolves to reach any position between the points illustrated in Fig. 3. So a grey wolf can update its position inside the space around the prey in any random location by using equations (3.1) and (3.2).

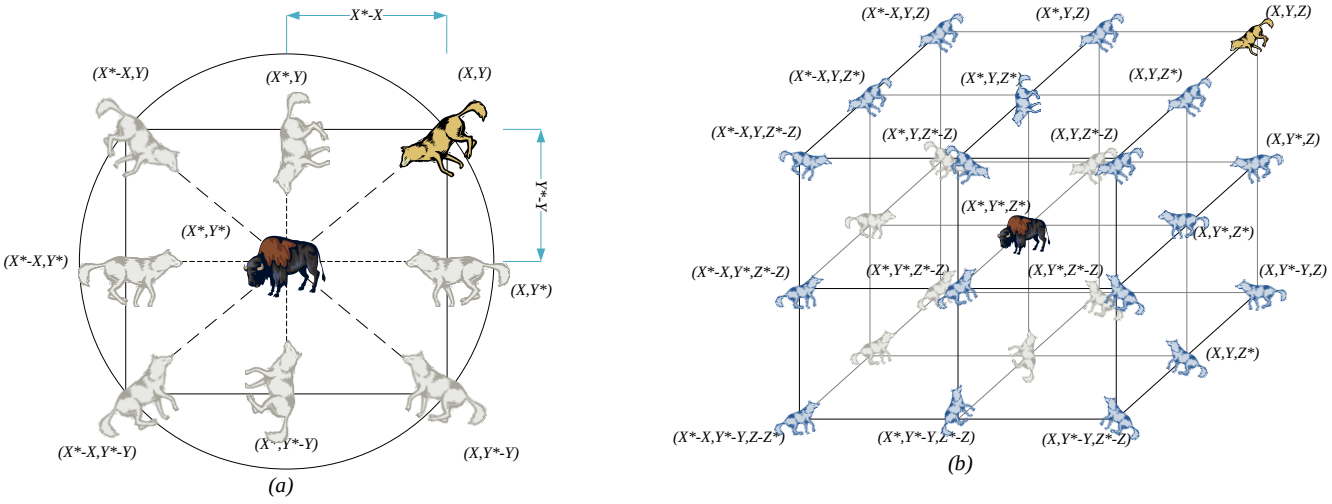


Fig. 3. *2D and 3D position vectors and their possible next locations*

The same concept can be extended to a search space with *n* dimensions, and the grey wolves will move in hyper-cubes (or hyper-spheres) around the best solution obtained so far.

3.2.3. Hunting:

Grey wolves have the ability to recognize the location of prey and encircle them. The hunt is usually guided by the alpha. The beta and delta might also participate in hunting occasionally. However, in an abstract search space we have no idea about the location of the optimum (prey). In order to mathematically simulate the hunting behavior of grey wolves, we suppose that the alpha (best candidate solution) beta, and delta have better knowledge about the potential location of prey. Therefore, we save the first three best solutions obtained so far and oblige the other search agents (including the omegas) to update their positions according to the position of the best search agent. The following formulas are proposed in this regard.

$$\overrightarrow{D_\alpha} = |\vec{C}_1.\overrightarrow{X_\alpha} - \vec{X}|, \overrightarrow{D_\beta} = |\vec{C}_2.\overrightarrow{X_\beta} - \vec{X}|, \overrightarrow{D_\delta} = |\overrightarrow{C_3}.\overrightarrow{X_\delta} - \vec{X}| \quad (3.5)$$

$$\overrightarrow{X_1} = \overrightarrow{X_\alpha} - \vec{A}_1.(\overrightarrow{D_\alpha}), \overrightarrow{X_2} = \overrightarrow{X_\beta} - \overrightarrow{A_2}.(\overrightarrow{D_\beta}), \overrightarrow{X_3} = \overrightarrow{X_\delta} - \overrightarrow{A_3}.(\overrightarrow{D_\delta}) \quad (3.6)$$

$$\vec{X}(t+1) = \frac{\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3}}{3} \quad (3.7)$$

Fig. 4 shows how a search agent updates its position according to alpha, beta, and delta in a 2D search space. It can be observed that the final position would be in a random place within a circle which is defined by the positions of alpha, beta, and delta in the search space. In other words alpha, beta, and delta estimate the position of the prey, and other wolves updates their positions randomly around the prey.

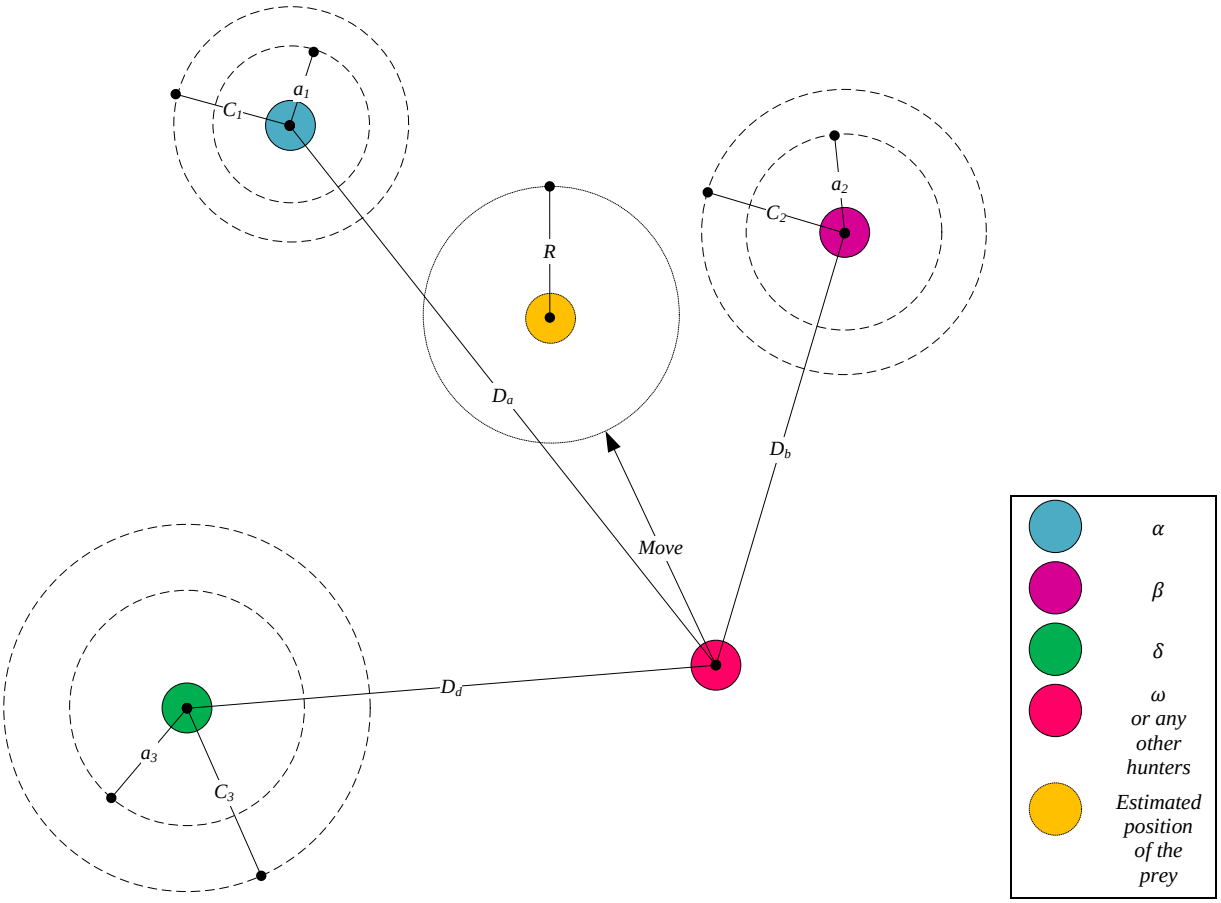
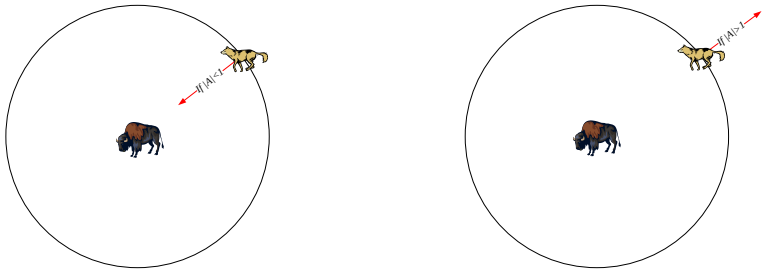


Fig. 4. *Position updading in GWO*

3.2.4. Attacking prey (exploitation):

As mentioned above the grey wolves finish the hunt by attacking the prey when it stops moving. In order to mathematically model approaching the prey we decrease the value of $\vec{a}$. Note that the fluctuation range of $\vec{A}$ is also decreased by $\vec{a}$. In other words $\vec{A}$ is a random value in the interval [-*a*,*a*] where *a* is decreased from 2 to 0 over the course of iterations. When random values of $\vec{A}$ are in [-1,1], the next position of a search agent can be in any position between its current position and the position of the prey. Fig. 5 (a) shows that |A|<1 forces the wolves to attack towards the prey.



(a) *(b)*

Fig. 5. *Attacking prey versus searching for prey*

With the operators proposed so far, the GWO algorithm allows its search agents to update their position based on the location of the alpha, beta, and delta; and attack towards the prey. However, the GWO algorithm is prone to stagnation in local solutions with these operators. It is true that the encircling mechanism proposed shows exploration to some extent, but GWO needs more operators to emphasize exploration.

3.2.5. Search for prey (exploration):

Grey wolves mostly search according to the position of the alpha, beta, and delta. They diverge from each other to search for prey and converge to attack prey. In order to mathematically model divergence, we utilize $\vec{A}$ with random values greater than 1 or less than -1 to oblige the search agent to diverge from the prey. This emphasizes exploration and allows the GWO algorithm to search globally. Fig. 5(b) also shows that |A|>1 forces the grey wolves to diverge from the prey to hopefully find a fitter prey. Another component of GWO that favors exploration is $\vec{C}$. As may be seen in Equation (3.4), the $\vec{C}$ vector contains random values in [0, 2]. This component provides random weights for prey in order to stochastically emphasize (*C>1*) or deemphasize (*C<1*) the effect of prey in defining the distance in Equation (3.1). This assists GWO to show a more random behavior throughout optimization, favoring exploration and local optima avoidance. It is worth mentioning here that *C* is not linearly decreased in contrast to *A*. We deliberately require *C* to provide random values at all times in order to emphasize exploration not only during initial iterations but also final iterations. This component is very helpful in case of local optima stagnation, especially in the final iterations.

The *C* vector can be also considered as the effect of obstacles to approaching prey in nature. Generally speaking, the obstacles in nature appear in the hunting paths of wolves and in fact prevent them from quickly and conveniently approaching prey. This is exactly what the vector *C* does. Depending on the position of a wolf, it can randomly give the prey a weight and make it harder and farther to reach for wolves, or vice versa.

To sum up, the search process starts with creating a random population of grey wolves (candidate solutions) in the GWO algorithm. Over the course of iterations, alpha, beta, and delta wolves estimate the probable position of the prey. Each candidate solution updates its distance from the prey. The parameter *a* is decreased from 2 to 0 in order to emphasize exploration and exploitation, respectively. Candidate solutions tend to diverge from the prey when $|\vec{A}|>1$ and converge towards the prey when $|\vec{A}|<1$. Finally, the GWO algorithm is terminated by the satisfaction of an end criterion.

The pseudo code of the GWO algorithm is presented in Fig. 6.

```
Initialize the grey wolf population Xi (i = 1, 2, ..., n)
Initialize a, A, and C
Calculate the fitness of each search agent
Xα=the best search agent
Xβ=the second best search agent
Xδ=the third best search agent
while (t < Max number of iterations)
    for each search agent
            Update the position of the current search agent by equation (3.7)
    end for
    Update a, A, and C
    Calculate the fitness of all search agents
    Update Xα, Xβ, and Xδ
    t=t+1
end while
return Xα
```

Fig. 6. *Pseudo code of the GWO algorithm*

To see how GWO is theoretically able to solve optimization problems, some points may be noted:

- The proposed social hierarchy assists GWO to save the best solutions obtained so far over the course of iteration
- The proposed encircling mechanism defines a circle-shaped neighborhood around the solutions which can be extended to higher dimensions as a hyper-sphere
- The random parameters A and C assist candidate solutions to have hyper-spheres with different random radii
- The proposed hunting method allows candidate solutions to locate the probable position of the prey
- Exploration and exploitation are guaranteed by the adaptive values of a and A
- The adaptive values of parameters a and A allow GWO to smoothly transition between exploration and exploitation
- With decreasing A, half of the iterations are devoted to exploration ($|A|\geq 1$) and the other half are dedicated to exploitation ($|A|<1$)
- The GWO has only two main parameters to be adjusted (a and C)

There are possibilities to integrate mutation and other evolutionary operators to mimic the whole life cycle of grey wolves. However, we have kept the GWO algorithm as simple as possible with the fewest operators to be adjusted. Such mechanisms are recommended for future work. The source codes of this algorithm can be found in http://www.alimirjalili.com/GWO.html and http://www.mathworks.com.au/matlabcentral/fileexchange/44974.

4. Results and discussion

In this section the GWO algorithm is benchmarked on 29 benchmark functions. The first 23 benchmark functions are the classical functions utilized by many researchers [16, 48-51]. Despite the simplicity, we have chosen these test functions to be able to compare our results to those of the current meta-heuristics. These benchmark functions are listed in Table 1, Table 2, and Table 3 where *Dim* indicates dimension of the function, *Range* is the boundary of the function's search space, and f_{min} is the optimum. The other test beds that we have chosen are six composite benchmark functions from a CEC 2005 special session [52]. These benchmark functions are the shifted, rotated, expanded, and combined variants of the classical functions which offer the greatest complexity among the current benchmark functions [53]. Tables 4 lists the CEC 2005 test functions, where *Dim* indicates dimension of the function, *Range* is the boundary of the function's search space, and f_{min} is the optimum. Fig. 7, Fig. 8, Fig. 9, and Fig. 10 illustrate the 2D versions of the benchmark functions used.

Generally speaking, the benchmark functions used are minimization functions and can be divided into four groups: unimodal, multimodal, fixed-dimension multimodal, and composite functions. Note that a detailed descriptions of the composite benchmark functions are available in the CEC 2005 technical report [52].

Table 1. Unimodal benchmark functions

Function	Dim	Range	f_{min}
$f_1(x)=\sum_{i=1}^{n} x_i^2$	30	[-100,100]	0
$f_2(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	30	[-10,10]	0
$f_3(x)=\sum_{i=1}^{n}\left(\sum_{j-1}^{i} x_j\right)^2$	30	[-100,100]	0
$f_4(x)=\max_i\{\lvert x_i\rvert, 1\le i\le n\}$	30	[-100,100]	0
$f_5(x)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	30	[-30,30]	0
$f_6(x)=\sum_{i=1}^{n}([x_i+0.5])^2$	30	[-100,100]	0
$f_7(x)=\sum_{i=1}^{n} ix_i^4+random[0,1)$	30	[-1.28,1.28]	0

Table 2. Multimodal benchmark functions

Function	Dim	Range	f_{min}
$F_8(x)=\sum_{i=1}^{n}-x_i sin\left(\sqrt{\lvert x_i\rvert}\right)$	30	[-500,500]	-418.9829×5
$F_9(x)=\sum_{i=1}^{n}[x_i^2-10cos(2\pi x_i)+10]$	30	[-5.12,5.12]	0
$F_{10}(x)=-20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-exp\left(\frac{1}{n}\sum_{i=1}^{n}cos(2\pi x_i)\right)+20+e$	30	[-32,32]	0
$F_{11}(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	30	[-600,600]	0
$F_{12}(x)=$ $\frac{\pi}{n}\{10sin(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10sin^2(\pi y_{i+1})]+(y_n-1)^2\}+\sum_{i=1}^{n}u(x_i,10,100,4)$ $y_i=1+\frac{x_i+1}{4}$ $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m & x_i>a\\ 0 & -a<x_i<a\\ k(-x_i-a)^m & x_i<-a\end{cases}$	30	[-50,50]	0
$F_{13}(x)=0.1\{sin^2(3\pi x_1)+\sum_{i=1}^{n}(x_i-1)^2[1+sin^2(3\pi x_i+1)]+(x_n-1)^2[1+sin^2(2\pi x_n)]\}+\sum_{i=1}^{n}u(x_i,5,100,4)$	30	[-50,50]	0
$F_{14}(x)=-\sum_{i=1}^{n}\sin(x_i).\left(\sin\left(\frac{ix_i^2}{\pi}\right)\right)^{2m}, \quad m=10$	30	[0,π]	-4.687
$F_{15}(x)=\left[e^{-\sum_{i=1}^{n}\left(x_i/\beta\right)^{2m}}-2e^{-\sum_{i=1}^{n}x_i^2}\right].\prod_{i=1}^{n}\cos^2 x_i, \quad m=5$	30	[-20,20]	-1
$F_{16}(x)=\{[\sum_{i=1}^{n}\sin^2(x_i)]-exp(-\sum_{i=1}^{n}x_i^2)\}.exp\left[-\sum_{i=1}^{n}\sin^2\sqrt{\lvert x_i\rvert}\right]$	30	[-10,10]	-1

Table 3. Fixed-dimenstion multimodal benchmark functions

Function	Dim	Range	f_{min}
$F_{14}(x)=\left(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right)^{-1}$	2	[-65,65]	1
$F_{15}(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}\right]^2$	4	[-5,5]	0.00030
$F_{16}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	2	[-5,5]	-1.0316
$F_{17}(x)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)cosx_1+10$	2	[-5,5]	0.398
$F_{18}(x)=[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]\times[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	2	[-2,2]	3
$F_{19}(x)=-\sum_{i=1}^{4}c_i exp\left(-\sum_{j=1}^{3}a_{ij}\left(x_j-p_{ij}\right)^2\right)$	3	[1,3]	-3.86
$F_{20}(x)=-\sum_{i=1}^{4}c_i exp\left(-\sum_{j=1}^{6}a_{ij}\left(x_j-p_{ij}\right)^2\right)$	6	[0,1]	-3.32
$F_{21}(x)=-\sum_{i=1}^{5}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	4	[0,10]	-10.1532
$F_{22}(x)=-\sum_{i=1}^{7}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	4	[0,10]	-10.4028
$F_{23}(x)=-\sum_{i=1}^{10}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	4	[0,10]	-10.5363

Table 4. Composite benchmark functions

Function	Dim	Range	f_{min}
F_{24}(CF1): $f_1, f_2, f_3, \dots, f_{10}$ = Sphere Function $[б_1, б_2, б_3, \dots, б_{10}] = [1,1,1,..,1]$ $[\lambda_1, \lambda_2, \lambda_3 \dots, \lambda_{10}] = [5/100,5/100,5/100,..,5/100]$	10	[-5,5]	0
F_{25}(CF2): $f_1, f_2, f_3, \dots, f_{10} = Griewank's\ Function$ $[б_1, б_2, б_3, \dots, б_{10}] = [1,1,1,..,1]$ $[\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_{10}] = [5/100,5/100,5/100,..,5/100]$	10	[-5,5]	0
F_{26}(CF3): $f_1, f_2, f_3, \dots, f_{10} = Griewank's\ Function$ $[б_1, б_2, б_3, \dots, б_{10}] = [1,1,1,..,1]$ $[\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_{10}] = [1,1,1,..,1]$	10	[-5,5]	0
F_{27}(CF4): f_1, f_2 = Ackley'sFunction f_3, f_4 = Rastrigin's Function f_5, f_6 = Weierstrass Function f_7, f_8 = Griewank's Function f_9, f_{10} = Sphere Function $[б_1, б_2, б_3, \dots, б_{10}] = [1,1,1,..,1]$ $[\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_{10}] = [5/32, 5/32, 1, 1, 5/0.5, 5/0.5, 5/100, 5/100,$ $5/100, \quad 5/100]$	10	[-5,5]	0
F_{28}(CF5): f_1, f_2 = Rastrigin's Function f_3, f_4 = Weierstrass Function f_5, f_6 = Griewank's Function f_7, f_8 = Ackley'sFunction f_9, f_{10} = Sphere Function $[б_1, б_2, б_3, \dots, б_{10}] = [1,1,1,..,1]$ $[\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_{10}] = [1/5, 1/5, 5/0.5, 5/0.5, 5/100, 5/100, 5/32, 5/32,$ $5/100, \quad 5/100]$	10	[-5,5]	0
f_{29}(CF6): f_1, f_2 = Rastrigin's Function f_3, f_4 = Weierstrass Function f_5, f_6 = Griewank's Function f_7, f_8 = Ackley'sFunction f_9, f_{10} = Sphere Function $[б_1, б_2, б_3, \dots, б_{10}] = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$ $[\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_{10}] = [0.1 * 1/5, 0.2 * 1/5, 0.3 * 5/0.5, 0.4 * 5/0.5, 0.5 * 5/100,$ $0.6 * 5/100, 0.7 * 5/32, 0.8 * 5/32, 0.9 * 5/100, 1 * 5/100]$	10	[-5,5]	0

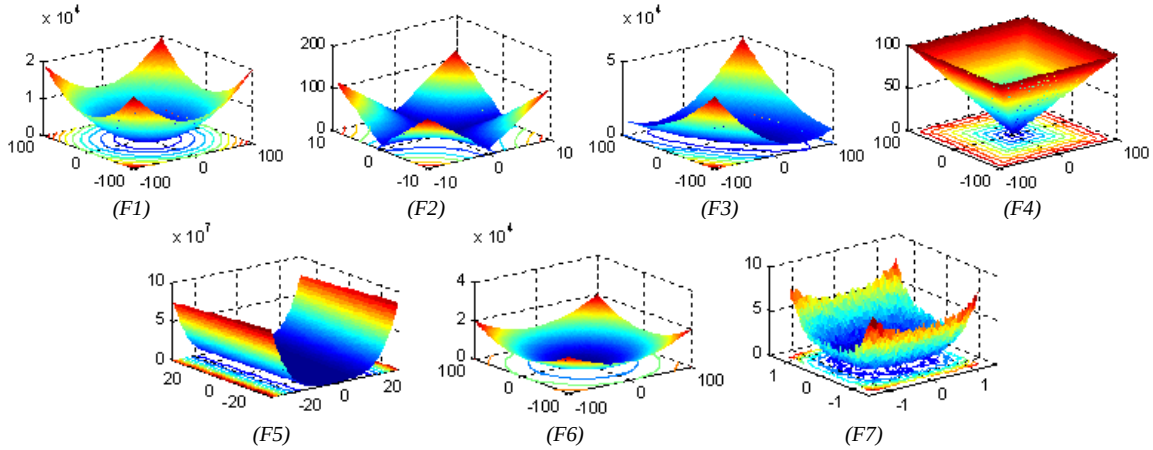


Fig. 7. 2-D versions of unimodal benchmark functions

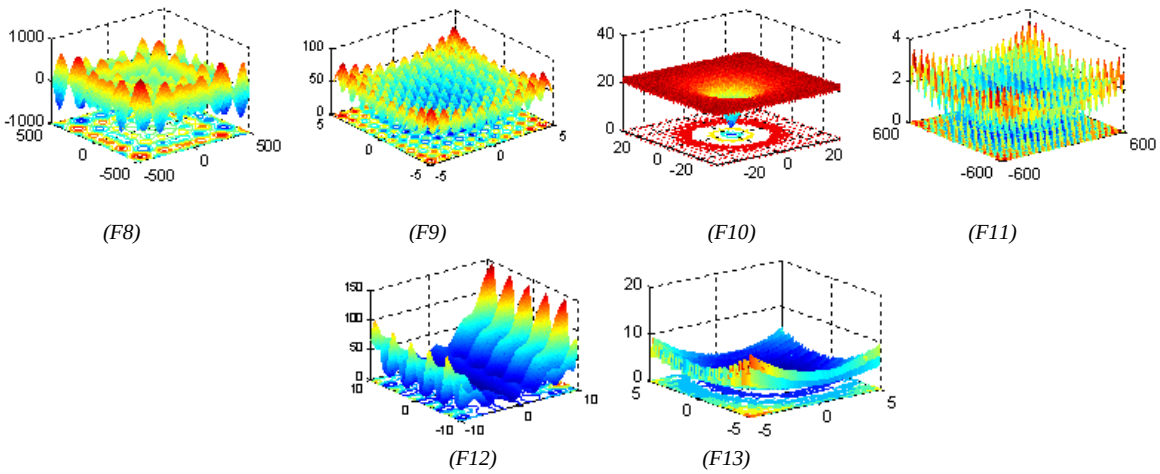


Fig. 8. 2-D versions of multimodal benchmark functions

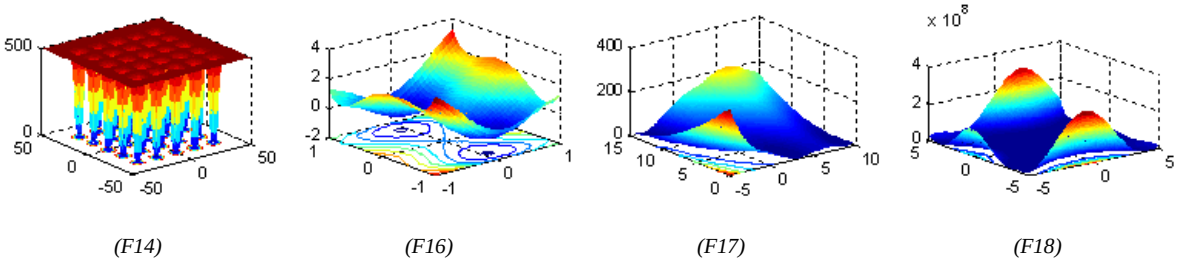


Fig. 9. 2-D version of fixed-dimension multimodal benchmark functions

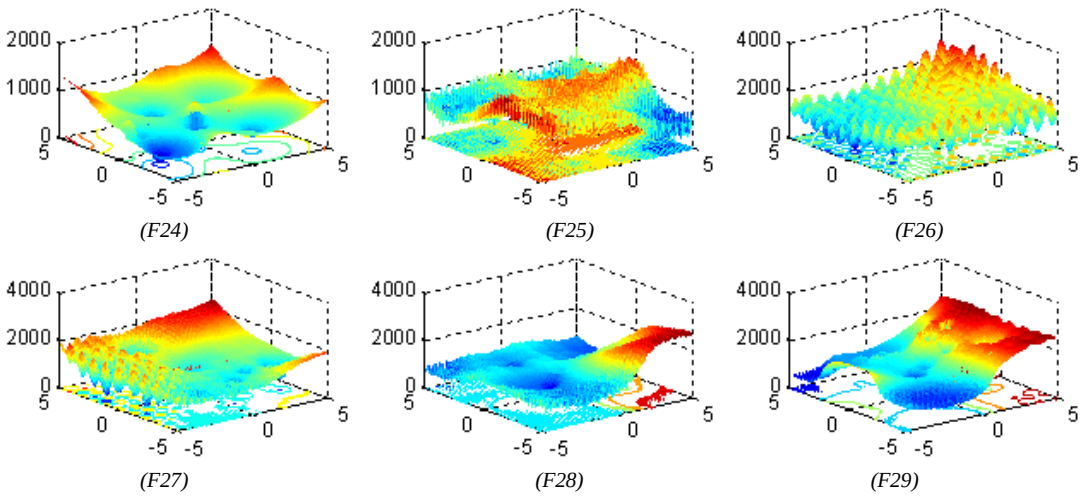


Fig. 10. 2-D versions of composite benchmark functions

The GWO algorithm was run 30 times on each benchmark function. The statistical results (average and standard deviation) are reported in Table 5 to Table 8. For verifying the results, the GWO algorithm is compared to PSO [3] as an SI-based technique and GSA [24] as a physics-based algorithm. In addition, the GWO algorithm is compared with three EAs: DE [15], Fast Evolutionary Programing (FEP) [16], and Evolution Strategy with Covariance Matrix Adaptation (CMA-ES) [18].

Table 5. Results of unimodal benchmark functions

F	GWO		PSO		GSA		DE		FEP	
	ave	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*
F1	6.59E-28	6.34E-05	0.000136	0.000202	2.53E-16	9.67E-17	8.2E-14	5.9E-14	0.00057	0.00013
F2	7.18E-17	0.029014	0.042144	0.045421	0.055655	0.194074	1.5E-09	9.9E-10	0.0081	0.00077
F3	3.29E-06	79.14958	70.12562	22.11924	896.5347	318.9559	6.8E-11	7.4E-11	0.016	0.014
F4	5.61E-07	1.315088	1.086481	0.317039	7.35487	1.741452	0	0	0.3	0.5
F5	26.81258	69.90499	96.71832	60.11559	67.54309	62.22534	0	0	5.06	5.87
F6	0.816579	0.000126	0.000102	8.28E-05	2.5E-16	1.74E-16	0	0	0	0
F7	0.002213	0.100286	0.122854	0.044957	0.089441	0.04339	0.00463	0.0012	0.1415	0.3522

Table 6. Results of multimodal benchmark functions

F	GWO		PSO		GSA		DE		FEP	
	ave	std	ave	std	ave	std	ave	std	ave	std
F8	-6123.1	-4087.44	-4841.29	1152.814	-2821.07	493.0375	-11080.1	574.7	-12554.5	52.6
F9	0.310521	47.35612	46.70423	11.62938	25.96841	7.470068	69.2	38.8	0.046	0.012
F10	1.06E-13	0.077835	0.276015	0.50901	0.062087	0.23628	9.7E-08	4.2E-08	0.018	0.0021
F11	0.004485	0.006659	0.009215	0.007724	27.70154	5.040343	0	0	0.016	0.022
F12	0.053438	0.020734	0.006917	0.026301	1.799617	0.95114	7.9E-15	8E-15	9.2E-06	3.6E-06
F13	0.654464	0.004474	0.006675	0.008907	8.899084	7.126241	5.1E-14	4.8E-14	0.00016	0.000073

Table 7. Results of fixed-dimension multimodal benchmark functions

F	GWO		PSO		GSA		DE		FEP	
	ave	std	ave	std	ave	std	ave	std	ave	std
F14	4.042493	4.252799	3.627168	2.560828	5.859838	3.831299	0.998004	3.3E-16	1.22	0.56
F15	0.000337	0.000625	0.000577	0.000222	0.003673	0.001647	4.5E-14	0.00033	0.0005	0.00032
F16	-1.03163	-1.03163	-1.03163	6.25E-16	-1.03163	4.88E-16	-1.03163	3.1E-13	-1.03	4.9E-07
F17	0.397889	0.397887	0.397887	0	0.397887	0	0.397887	9.9E-09	0.398	1.5E-07
F18	3.000028	3	3	1.33E-15	3	4.17E-15	3	2E-15	3.02	0.11
F19	-3.86263	-3.86278	-3.86278	2.58E-15	-3.86278	2.29E-15	N/A	N/A	-3.86	0.000014
F20	-3.28654	-3.25056	-3.26634	0.060516	-3.31778	0.023081	N/A	N/A	-3.27	0.059
F21	-10.1514	-9.14015	-6.8651	3.019644	-5.95512	3.737079	-10.1532	0.0000025	-5.52	1.59
F22	-10.4015	-8.58441	-8.45653	3.087094	-9.68447	2.014088	-10.4029	3.9E-07	-5.53	2.12
F23	-10.5343	-8.55899	-9.95291	1.782786	-10.5364	2.6E-15	-10.5364	1.9E-07	-6.57	3.14

Table 8. Results of composite benchmark functions

F	GWO		PSO		GSA		DE		CMA-ES	
	ave	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*
F24	43.83544	69.86146	100	81.65	6.63E-17	2.78E-17	6.75E-02	1.11E-01	100	188.56
F25	91.80086	95.5518	155.91	13.176	200.6202	67.72087	28.759	8.6277	161.99	151
F26	61.43776	68.68816	172.03	32.769	180	91.89366	144.41	19.401	214.06	74.181
F27	123.1235	163.9937	314.3	20.066	170	82.32726	324.86	14.784	616.4	671.92
F28	102.1429	81.25536	83.45	101.11	200	47.14045	10.789	2.604	358.3	168.26
F29	43.14261	84.48573	861.42	125.81	142.0906	88.87141	490.94	39.461	900.26	8.32E-02

4.1. Exploitation analysis:

According to the results of Table 5, GWO is able to provide very competitive results. This algorithm outperforms all others in F1, F2, and F7. It may be noted that the unimodal functions are suitable for benchmarking exploitation. Therefore, these results show the superior performance of GWO in terms of exploiting the optimum. This is due to the proposed exploitation operators previously discussed.

4.2. Exploration analysis:

In contrast to the unimodal functions, multimodal functions have many local optima with the number increasing exponentially with dimension. This makes them suitable for benchmarking the exploration ability of an algorithm. According to the results of Table 6 and Table 7, GWO is able to provide very competitive results

on the multimodal benchmark functions as well. This algorithm outperforms PSO and GSA on the majority of the multimodal functions. Moreover, GWO shows very competitive results compare to DE and FEP; and outperforms them occasionally. These results show that the GWO algorithm has merit in terms of exploration.

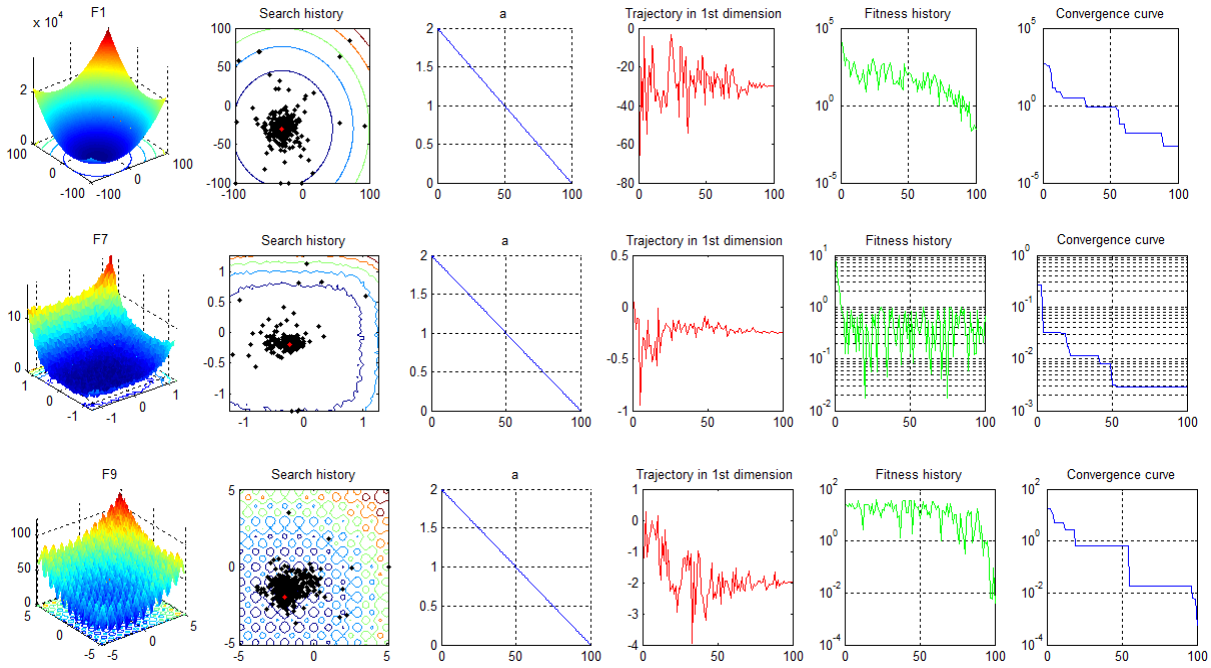
4.3. Local minima avoidance

The fourth class of benchmark functions employed includes composite functions, generally very challenging test beds for meta-heuristic algorithms. So, exploration and exploitation can be simultaneously benchmarked by the composite functions. Moreover, the local optima avoidance of an algorithm can be examined due to the massive number of local optima in such test functions. According to Table 8, GWO outperforms all others on half of the composite benchmark functions. The GWO algorithm also provides very competitive results on the remaining composite benchmark functions. This demonstrates that GWO shows a good balance between exploration and exploitation that results in high local optima avoidance. This superior capability is due to the adaptive value of A. As mentioned above, half of the iterations are devoted to exploration ($|A|\geq 1$) and the rest to exploitation ($|A|<1$). This mechanism assists GWO to provide very good exploration, local minima avoidance, and exploitation simultaneously.

4.4. Convergence behavior analysis

In this subsection the convergence behavior of GWO is investigated. According to Berg *et al.* [54], there should be abrupt changes in the movement of search agents over the initial steps of optimization. This assists a meta-heuristic to explore the search space extensively. Then, these changes should be reduced to emphasize exploitation at the end of optimization. In order to observe the convergence behavior of the GWO algorithm, the search history and trajectory of the first search agent in its first dimension are illustrated in Fig. 11. The animated versions of this figure can be found in supplementary materials. Note that the benchmark functions are shifted in this section, and we used six search agents to find the optima.

The second column of Fig. 11 depicts the search history of the search agents. It may be observed that the search agents of GWO tend to extensively search promising regions of the search spaces and exploit the best one. In addition, the fourth column of Fig. 11 shows the trajectory of the first particle, in which changes of the first search agent in its first dimension can be observed. It can be seen that there are abrupt changes in the initial steps of iterations which are decreased gradually over the course of iterations. According to Berg *et al.* [54], this behavior can guarantee that a SI algorithm eventually convergences to a point in search space.



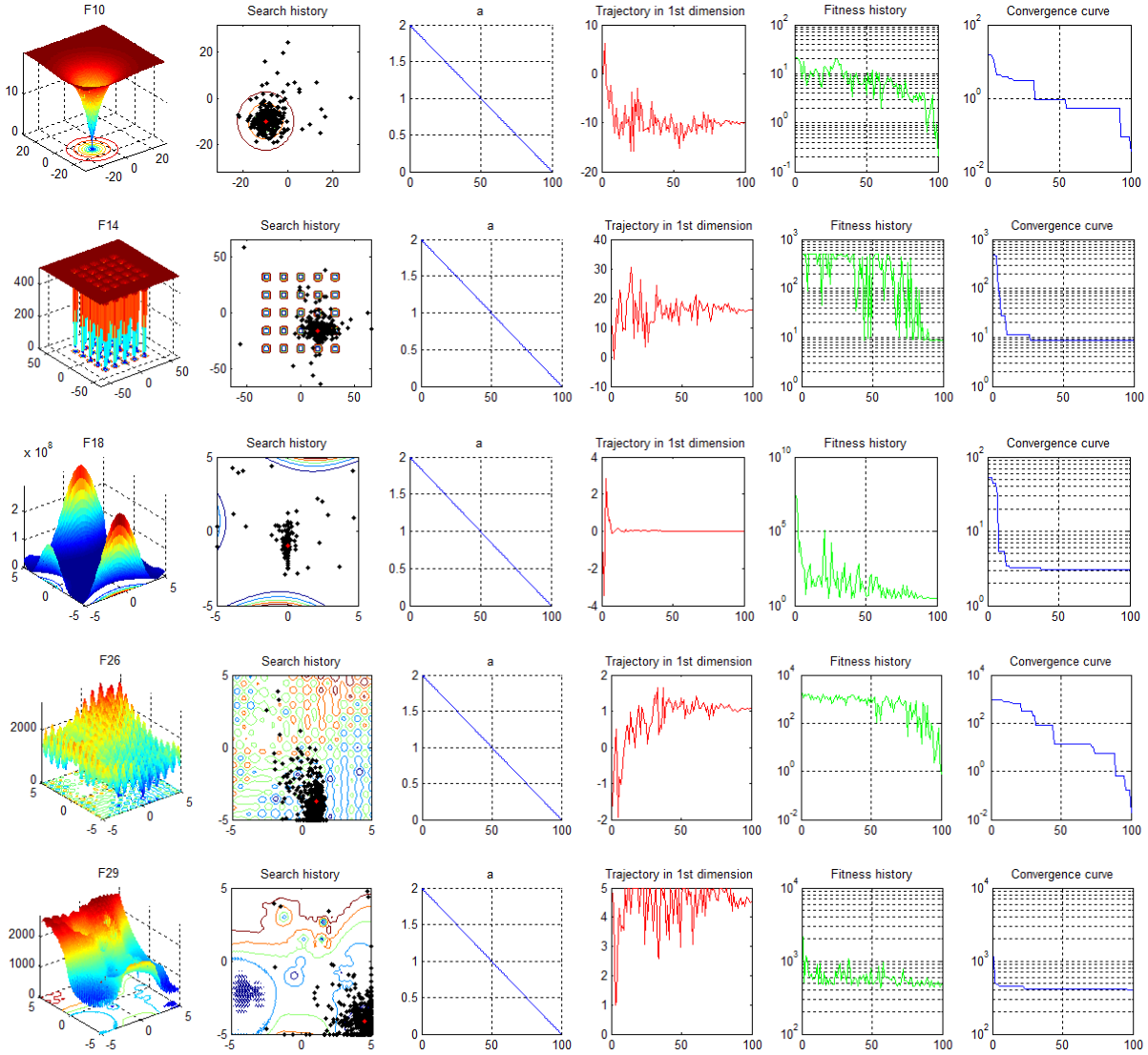


Fig. 11. Search history and trajectory of the first particle in the first dimension

To sum up, the results verify the performance of the GWO algorithm in solving various benchmark functions compared to well-known meta-heuristics. To further investigate the performance of the proposed algorithm, three classical engineering design problems and a real problem in optical engineering are employed in the following sections. The GWO algorithm is also compared with well-known techniques to confirm its results.

5. GWO for classical engineering problems

In this section three constrained engineering design problems: tension/compression spring, welded beam, and pressure vessel designs, are employed. These problems have several equality and inequality constraints, so the GWO should be equipped with a constraint handling method to be able to optimize constrained problems as well. Generally speaking, constraint handling becomes very challenging when the fitness function directly affects the position updating of the search agents (GSA for instance). For the fitness independent algorithms, however, any kind of constraint handling can be employed without the need to modify the mechanism of the algorithm (GA and PSO for instance). Since the search agents of the proposed GWO algorithm update their positions with respect to the alpha, beta, and delta locations, there is no direct relation between the search agents and the fitness function. So the simplest constraint handling method, penalty functions, where search agents are assigned big objective function values if they violate any of the constraints, can be employed effectively to handle constraints in GWO. In this case, if the alpha, beta, or delta violate constraints, they are automatically replaced with a new search agent in the next iteration. Any kind of penalty function can readily be employed in order to penalize search agents based on their level of violation. In this case, if the penalty makes the alpha,

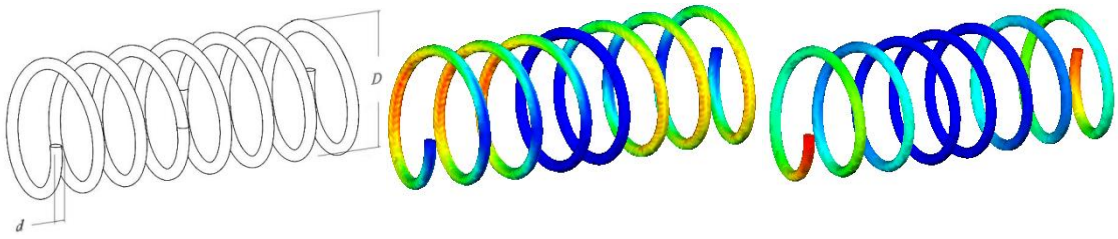
beta, or delta less fit than any other wolves, it is automatically replaced with a new search agent in the next iteration. We used simple, scalar penalty functions for the rest of problems except the tension/compression spring design problem which uses a more complex penalty function.

5.1. Tension/compression spring design

The objective of this problem is to minimize the weight of a tension/compression spring as illustrated in Fig. 12 [55-57]. The minimization process is subject to some constraints such as shear stress, surge frequency, and minimum deflection. There are three variables in this problem: wire diameter (d), mean coil diameter (D), and the number of active coils (N). The mathematical formulation of this problem is as follows:

$$
\begin{aligned}
&\text{Consider} && \vec{x} = [x_1\ x_2\ x_3] = [d\ D\ N],\\
&\text{Minimize} && f(\vec{x}) = (x_3 + 2)x_2x_1^2,\\
&\text{Subject to} && g_1(\vec{x}) = 1 - \frac{x_2^3 x_3}{71785x_1^4} \le 0,\\
& && g_2(\vec{x}) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} \le 0,\\
& && g_3(\vec{x}) = 1 - \frac{140.45x_1}{x_2^2x_3} \le 0,\\
& && g_4(\vec{x}) = \frac{x_1 + x_2}{1.5} - 1 \le 0,\\
&\text{Variable range} && 0.05 \le x_1 \le 2.00,\\
& && 0.25 \le x_2 \le 1.30,\\
& && 2.00 \le x_3 \le 15.0
\end{aligned}
\tag{5.1}
$$

This problem has been tackled by both mathematical and heuristic approaches. Ha and Wang tried to solve this problem using PSO [58]. The Evolution Strategy (ES) [59], GA [60], Harmony Search (HS) [61], and Differential Evolution (DE) [62] algorithms have also been employed as heuristic optimizers for this problem. The mathematical approaches that have been adopted to solve this problem are the numerical optimization technique (constraints correction at constant cost) [55] and mathematical optimization technique [56]. The comparison of results of these techniques and GWO are provided in Table 9. Note that we use a similar penalty function for GWO to perform a fair comparison [63]. Table 9 suggests that GWO finds a design with the minimum weight for this problem.



(a) (b) (c)

Fig. 12. Tension/compression spring: (a) shematic, (b) stress heatmap (c) displacement heatmap

Table 9. Comparison of results for tension/compression spring design problem

Algorithm	Optimum variables			Optimum weight
	d	*D*	*N*	
GWO	0.05169	0.356737	11.28885	0.012666
GSA	0.050276	0.323680	13.525410	0.0127022
PSO (Ha and Wang)	0.051728	0.357644	11.244543	0.0126747
ES (Coello and Montes)	0.051989	0.363965	10.890522	0.0126810
GA (Coello)	0.051480	0.351661	11.632201	0.0127048
HS (Mahdavi et al.)	0.051154	0.349871	12.076432	0.0126706
DE (Huang et al.)	0.051609	0.354714	11.410831	0.0126702
Mathematical optimization (Belegundu)	0.053396	0.399180	9.1854000	0.0127303
Constraint correction (Arora)	0.050000	0.315900	14.250000	0.0128334

5.2. Welded beam design:

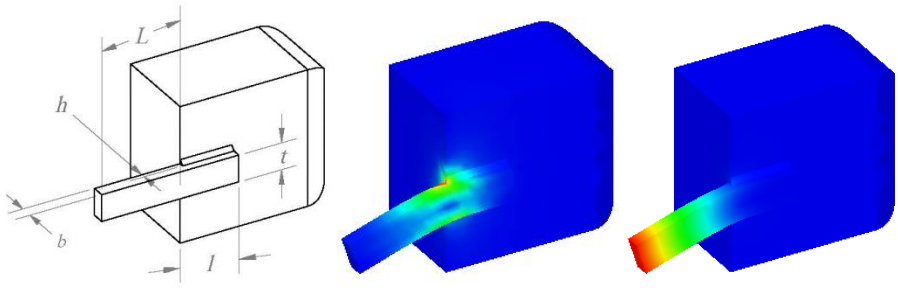
The objective of this problem is to minimize the fabrication cost of a welded beam as shown in Fig. 13 [60]. The constraints are as follows:

- Shear stress (τ)
- Bending stress in the beam (θ)
- Buckling load on the bar (P_c)
- End deflection of the beam (δ)
- Side constraints.

This problem has four variables such as thickness of weld (h), length of attached part of bar (l), the height of the bar (t), and thickness of the bar (b). The mathematical formulation is as follows:

$$
\begin{aligned}
&\text{Consider} && \vec{x} = [x_1\ x_2\ x_3\ x_4] = [h\ l\ t\ b],\\
&\text{Minimize} && f(\vec{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2),\\
&\text{Subject to} && g_1(\vec{x}) = \tau(\vec{x}) - \tau_{max} \le 0,\\
& && g_2(\vec{x}) = \sigma(\vec{x}) - \sigma_{max} \le 0,\\
& && g_3(\vec{x}) = \delta(\vec{x}) - \delta_{max} \le 0,\\
& && g_4(\vec{x}) = x_1 - x_4 \le 0,\\
& && g_5(\vec{x}) = P - P_c(\vec{x}) \le 0,\\
& && g_6(\vec{x}) = 0.125 - x_1 \le 0\\
& && g_7(\vec{x}) = 1.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \le 0 \qquad (5.2)\\
&\text{Variable range} && 0.1 \le x_1 \le 2,\\
& && 0.1 \le x_2 \le 10,\\
& && 0.1 \le x_3 \le 10,\\
& && 0.1 \le x_4 \le 2\\
&\text{where} && \tau(\vec{x}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2},\\
& && \tau' = \frac{P}{\sqrt{2}x_1x_2},\ \tau'' = \frac{MR}{J}, M = P\left(L + \frac{x_2}{2}\right),\\
& && R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2},\\
& && J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\},\\
& && \sigma(\vec{x}) = \frac{6PL}{x_4x_3^2}, \delta(\vec{x}) = \frac{6PL^3}{Ex_3^2x_4}\\
& && P_c(\vec{x}) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right),\\
& && P = 6000\ \text{lb}, L = 14\ \text{in.}, \delta_{max} = 0.25\ \text{in.}, E = 30 \times 1^6\ \text{psi}, G = 12 \times 10^6\ \text{psi},\\
& && \tau_{max} = 13600\ \text{psi}, \sigma_{max} = 30000\ \text{psi}
\end{aligned}
$$

Coello [64] and Deb [65, 66] employed GA, whereas Lee and Geem [67] used HS to solve this problem. Richardson's random method, Simplex method, Davidon-Fletcher-Powell, Griffith and Stewart's successive linear approximation are the mathematical approaches that have been adopted by Ragsdell and Philips [68] for this problem. The comparison results are provided in Table 10. The results show that GWO finds a design with the minimum cost compared to others.



(a) (b) (c)

Fig. 13. Structure of welded beam design (a) shematic (b) stress heatmap (c) displacement heatmap

Table 10. Comparison results of the welded beam design problem

Algorithm	Optimum variables				Optimum cost
	h	*l*	*t*	*b*	
GWO	0.205676	3.478377	9.03681	0.205778	1.72624
GSA	0.182129	3.856979	10.00000	0.202376	1.879952
GA Coello)	N/A	N/A	N/A	N/A	1.8245
GA (Deb)	N/A	N/A	N/A	N/A	2.3800
GA (Deb)	0.2489	6.1730	8.1789	0.2533	2.4331
HS (Lee and Geem)	0.2442	6.2231	8.2915	0.2443	2.3807
Random	0.4575	4.7313	5.0853	0.6600	4.1185
Simplex	0.2792	5.6256	7.7512	0.2796	2.5307
David	0.2434	6.2552	8.2915	0.2444	2.3841
APPROX	0.2444	6.2189	8.2915	0.2444	2.3815

5.3. Pressure vessel design:

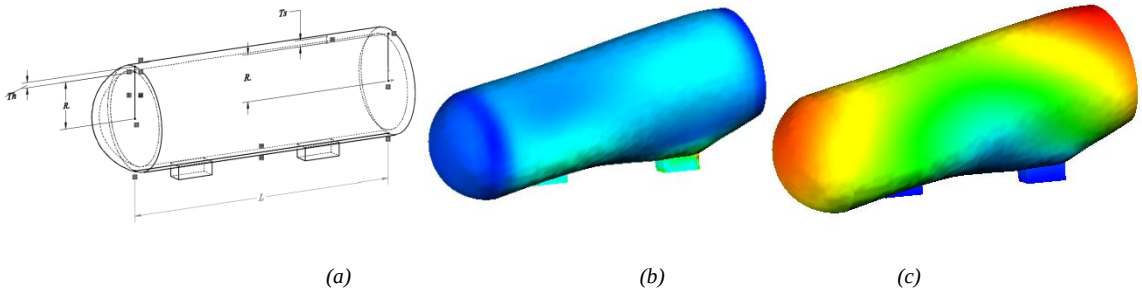
The objective of this problem is to minimize the total cost consisting of material, forming, and welding of a cylindrical vessel as in Fig. 14. Both ends of the vessel are capped, and the head has a hemi-spherical shape. There are four variables in this problem:

- Thickness of the shell (T_s)
- Thickness of the head (T_h)
- Inner radius (R)
- Length of the cylindrical section without considering the head (L)

This problem is subject to four constraints. These constraints and the problem are formulated as follows:

$$
\begin{aligned}
&\text{Consider} && \vec{x} = [x_1\ x_2\ x_3\ x_4] = [T_s\ T_h\ R\ L],\\
&\text{Minimize} && f(\vec{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3,\\
&\text{Subject to} && g_1(\vec{x}) = -x_1 + 0.0193x_3 \le 0,\\
& && g_2(\vec{x}) = -x_3 + 0.00954x_3 \le 0,\\
& && g_3(\vec{x}) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \le 0,\\
& && g_4(\vec{x}) = x_4 - 240 \le 0,\\
&\text{Variable range} && 0 \le x_1 \le 99,\\
& && 0 \le x_2 \le 99,\\
& && 10 \le x_3 \le 200,\\
& && 10 \le x_4 \le 200
\end{aligned}
\qquad (5.3)
$$

This problem has also been popular among researchers and optimized in various studies. The heuristic methods that have been adopted to optimize this problem are: PSO [58], GA [57, 60, 69], ES [59], DE [62], and ACO [70]. Mathematical methods used are augmented Lagrangian Multiplier [71] and branch-and-bound [72]. The results of this problem are provided in Table 11. According to this table, GWO is again able to find a design with the minimum cost.



(a) *(b)* *(c)*

Fig. 14. Pressure vessel (a) shematic (b) stress heatmap (c) displacement heatmap

Table 11. Comparison results for pressure vessel design problem

Algorithm	Optimum variables				Optimum cost
	T_s	T_h	R	L	
GWO	0.812500	0.434500	42.089181	176.758731	6051.5639
GSA	1.125000	0.625000	55.9886598	84.4542025	8538.8359
PSO (He and Wang)	0.812500	0.437500	42.091266	176.746500	6061.0777
GA (Coello)	0.812500	0.434500	40.323900	200.000000	6288.7445
GA (Coello and Montes)	0.812500	0.437500	42.097398	176.654050	6059.9463
GA (Deb and Gene)	0.937500	0.500000	48.329000	112.679000	6410.3811
ES (Montes and Coello)	0.812500	0.437500	42.098087	176.640518	6059.7456
DE (Huang et al.)	0.812500	0.437500	42.098411	176.637690	6059.7340
ACO (Kaveh and Talataheri)	0.812500	0.437500	42.103624	176.572656	6059.0888
Lagrangian Multiplier (Kannan)	1.125000	0.625000	58.291000	43.6900000	7198.0428
branch-bound (Sandgren)	1.125000	0.625000	47.700000	117.701000	8129.1036

In summary, the results on the three classical engineering problems demonstrate that GWO shows high performance in solving challenging problems. This is again due to the operators that are designed to allow GWO to avoid local optima successfully and converge towards the optimum quickly. The next section probes the performance of the GWO algorithm in solving a recent real problem in the field of optical engineering.

6. Real application of GWO in optical engineering (optical buffer design)

The problem investigated in this section is called optical buffer design. In fact, an optical buffer is one of the main components of optical CPUs. The optical buffer slows the group velocity of light and allows the optical CPUs to process optical packets or adjust its timing. The most popular device to do this is a Photonic Crystal Waveguide (PCW). PCWs mostly have a lattice-shaped structure with a line defect in the middle. The radii of holes and shape of the line defect yield different slow light characteristics. Varying radii and line defects provides different environments for refracting the light in the waveguide. The researchers in this field try to manipulate the radii of holes and pins of line defect in order to achieve desirable optical buffering characteristics. There are also different types of PCW that are suitable for specific applications. In this section the structure of a PCW called a Bragg Slot PCW (BSPCW) is optimized by the GWO algorithm. This problem has several constraints, so we utilize the simplest constraint handling method for GWO in this section as well.

BSPCW structure was first proposed by C. Caer *et al.* in 2011 [73]. The structure of BSPCWs is illustrated in Fig. 15. The background slab is silicon with a refractive index equal to 3.48. The slot and holes are filled by a material with a refractive index of 1.6. The Bragg slot structure allows the BSPCW to have precise control of dispersion and slow light properties. The first five holes adjacent to the slot have the highest impact on slow light properties, as discussed in [73]. As may be seen in Fig. 15, l, w_l, and w_h define the shape of the slot and have an impact on the final dispersion and slow light properties as well. So, various dispersion and slow light properties can be achieved by manipulating the radii of holes, l, w_l, and w_h.

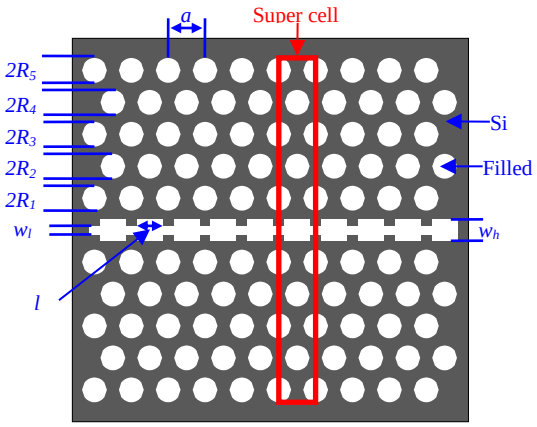


Fig. 15. BSPCW structure with super cell, $n_{background}$=3.48 and n_{filled}=1.6.

There are two metrics for comparing the performance of slow light devices: Delay-Bandwidth Product (DBP) and Normalized DBP (NDBP), which are defined as follows [74]:

$$DBP = \Delta t.\Delta f \tag{6.1}$$

where Δt indicates the delay and Δf is the bandwidth of the slow light device.

In slow light devices the ultimate goal is to achieve maximum transmission delay of an optical pulse with highest PCW bandwidth. Obviously, Δt should be increased in order to increase DBP. This is achieved by increasing the length of the device (L). To compare devices with different lengths and operating frequencies, NDBP is a better choice [75]:

$$\mathrm{NDBP} = \overline{\mathrm{n_g}}.\Delta\omega/\omega_0 \tag{6.2}$$

where $\overline{\mathrm{ng}}$ is the average of the group index , Δω is the normalized bandwidth, and ω_0 is the normalized central frequency of light wave.

Since NDBP has a direct relation to the group index (n_g), $n_g$can be formulated as follows [76]:

$$\mathrm{n_g} = \frac{\mathrm{C}}{\mathrm{v_g}} = \mathrm{C}\frac{\mathrm{dk}}{\mathrm{d}\omega} \tag{6.3}$$

where ω is the dispersion, k indicates the wave vector, C is the velocity of light in free space, and n_g shows the group index. Since n_g is changing in the bandwidth range, it should be averaged as follows:

$$\overline{\mathrm{n_g}} = \int_{\omega_L}^{\omega_H} \mathrm{n_g}(\omega)\frac{\mathrm{d}\omega}{\Delta\omega} \tag{6.4}$$

The bandwidth of a PCW refers to the region of the n_g curve where n_g has an approximately constant value with a maximum fluctuation rage of ±10% [75]. Detailed information about PCWs can be found in [77-80].

Finally, the problem is mathematically formulated for GWO as follows:

$$\begin{aligned}
&\text{Consider:} && \vec{\mathrm{x}} = [\mathrm{x_1}\ \mathrm{x_2}\ \mathrm{x_3}\ \mathrm{x_4}\ \mathrm{x_5}\ \mathrm{x_6}\ \mathrm{x_7}\ \mathrm{x_8}] = \left[\frac{\mathrm{R_1}}{\mathrm{a}}\ \frac{\mathrm{R_2}}{\mathrm{a}}\ \frac{\mathrm{R_3}}{\mathrm{a}}\ \frac{\mathrm{R_4}}{\mathrm{a}}\ \frac{\mathrm{R_5}}{\mathrm{a}}\ \frac{\mathrm{l}}{\mathrm{a}}\ \frac{\mathrm{w_h}}{\mathrm{a}}\ \frac{\mathrm{w_l}}{\mathrm{a}}\right],\\
&\text{Maximize:} && \mathrm{f}(\vec{\mathrm{x}}) = \mathrm{NDBP} = \frac{\overline{\mathrm{n_g}}\Delta\omega}{\omega_0},\\
&\text{Subject to:} && \max(|\beta_2(\omega)|) < 10^6\ \mathrm{a}/2\pi\mathrm{c}^2,\\
& && \omega_H < \min(\omega_{\text{up band}}),\\
& && \omega_L > \max(\omega_{\text{down band}}),\\
& && \mathrm{k_n} > \mathrm{k_{n\,H}} \rightarrow \omega_{\text{Guided mode}} > \omega_H,
\end{aligned} \tag{6.5}$$

$$k_n < k_{n\,L} \rightarrow \omega_{\text{Guided mode}} < \omega_L \ ,$$

where:

$$\omega_H = \omega(k_{n\,H}) = \omega(1.1 n_{g0}),$$
$$\omega_L = \omega(k_{n\,L}) = \omega(0.9 n_{g0}),$$
$$k_n = \frac{ka}{2\pi},$$
$$\Delta\omega = \omega_H - \omega_L,$$
$$a = \omega_0 * 1550 \text{ (nm)},$$

Variable range:

$$0 \le x_{1-5} \le 0.5,$$
$$0 \le x_6 \le 1,$$
$$0 \le x_{7,8} \le 1$$

Note that we consider five constraints for the GWO algorithm. The second to fifth constraints avoid band mixing. To handle feasibility, we assign small negative objective function values (-100) to those search agents that violate the constraints.

The GWO algorithm was run 20 times on this problem and the best results obtained are reported in Table 12. Note that the algorithm was run by 24 CPUs on a Windows HPC cluster at Griffith University. This table shows that there is a substantial, 93% and 65% improvement in bandwidth (Δλ) and NDBP utilizing the GWO algorithm.

The photonic band structure of the BSPCW optimized is shown in Fig. 16(a). In addition, the corresponded group index and optimized super cell are shown in Fig. 16 (b) and Fig. 17. These figures show that the optimized structure has a very good bandwidth without band mixing as well. This again demonstrated the high performance of the GWO algorithm in solving real problems.

Table 12. Structural parameters and calculation results

Structural parameter	Wu *et al.*[81]	GWO
R_1	-	0.33235a
R_2	-	0.24952a
R_3	-	0.26837a
R_4	-	0.29498a
R_5	-	0.34992a
l	-	0.7437a
W_h	-	0.2014a
W_l	-	0.60073a
a(nm)	430	343
$\overline{n_g}$	23	19.6
Δλ(nm)	17.6	33.9
Order of magnitude of β_2 ($a/2\pi c^2$)	10^3	10^3
NDBP	0.26	0.43

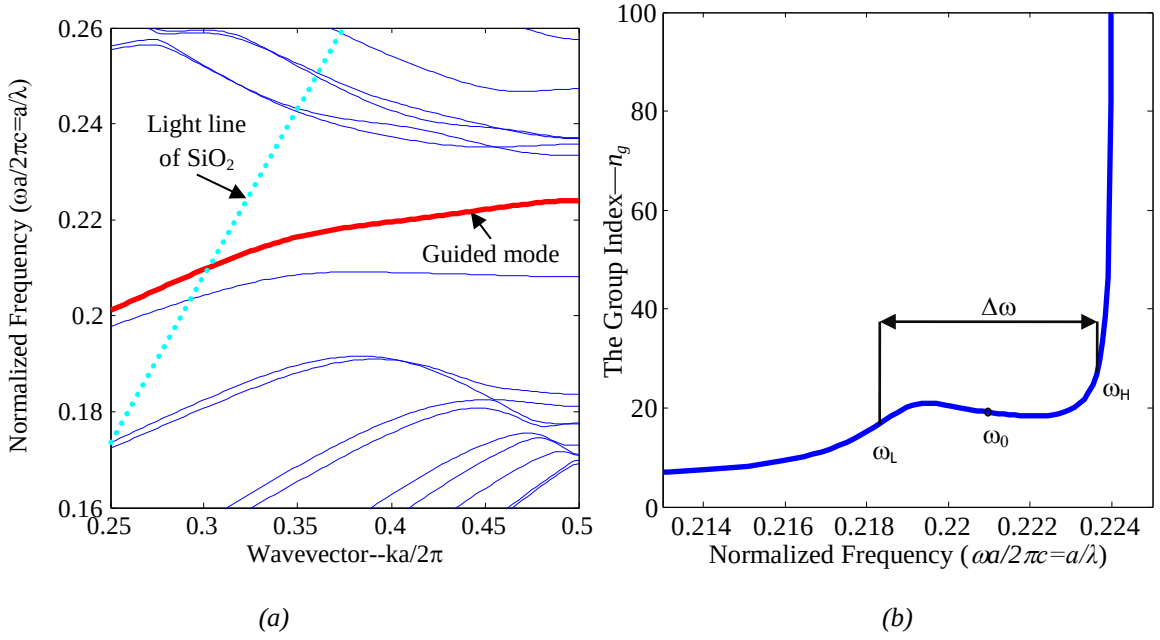


Fig. 16. (a) Photonic band structure of the optimized BSPCW structure (b) The group index (n_g) of the optimized BSPCW structure

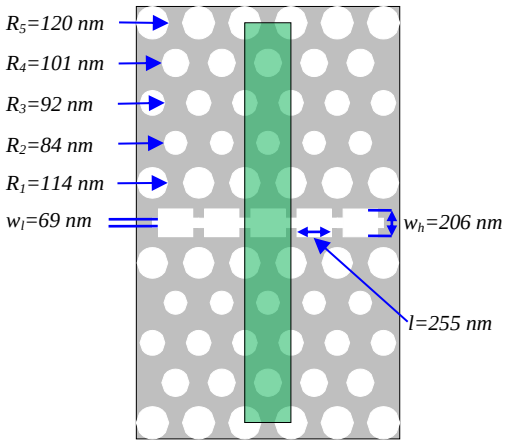


Fig. 17. Optimized super cell of BSPCW.

This comprehensive study shows that the proposed GWO algorithm has merit among the current meta-heuristics. First, the results of the unconstrained benchmark functions demonstrate the performance of the GWO algorithm in terms of exploration, exploitation, local optima avoidance, and convergence. Second, the results of the classical engineering problems show the superior performance of the proposed algorithm in solving semi-real constrained problems. Finally, the results of the optical buffer design problem show the ability of the GWO algorithm in solving the real problems.

7. Conclusion

This work proposed a novel SI optimization algorithm inspired by grey wolves. The proposed method mimicked the social hierarchy and hunting behavior of grey wolves. Twenty nine test functions were employed in order to benchmark the performance of the proposed algorithm in terms of exploration, exploitation, local optima avoidance, and convergence. The results showed that GWO was able to provide highly competitive results compared to well-known heuristics such as PSO, GSA, DE, EP, and ES. First, the results on the unimodal functions showed the superior exploitation of the GWO algorithm. Second, the exploration ability of GWO was confirmed by the results on multimodal functions. Third, the results of the composite functions showed high local optima avoidance. Finally, the convergence analysis of GWO confirmed the convergence of this algorithm.

Moreover, the results of the engineering design problems also showed that the GWO algorithm has high performance in unknown, challenging search spaces. The GWO algorithm was finally applied to a real problem in optical engineering. The results on this problem showed a substantial improvement of NDBP compared to current approaches, showing the applicability of the proposed algorithm in solving real problems. It may be noted that the results on semi-real and real problems also proved that GWO can show high performance not only on unconstrained problems but also on constrained problems.

For future work, we are going to develop binary and multi-objective versions of the GWO algorithm.

References

[1] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm intelligence: from natural to artificial systems*: OUP USA, 1999.

[2] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *Computational Intelligence Magazine, IEEE*, vol. 1, pp. 28-39, 2006.

[3] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, 1995, pp. 1942-1948.

[4] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 1, pp. 67-82, 1997.

[5] S. Kirkpatrick, D. G. Jr., and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, pp. 671-680, 1983.

[6] G. Beni and J. Wang, "Swarm intelligence in cellular robotic systems," in *Robots and Biological Systems: Towards a New Bionics?*, ed: Springer, 1993, pp. 703-712.

[7] B. Basturk and D. Karaboga, "An artificial bee colony (ABC) algorithm for numeric function optimization," in *IEEE swarm intelligence symposium*, 2006, pp. 12-14.

[8] O. Olorunda and A. P. Engelbrecht, "Measuring exploration/exploitation in particle swarms using swarm diversity," in *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, 2008, pp. 1128-1134.

[9] E. Alba and B. Dorronsoro, "The exploration/exploitation tradeoff in dynamic cellular genetic algorithms," *Evolutionary Computation, IEEE Transactions on*, vol. 9, pp. 126-142, 2005.

[10] L. Lin and M. Gen, "Auto-tuning strategy for evolutionary algorithms: balancing between exploration and exploitation," *Soft Computing*, vol. 13, pp. 157-168, 2009.
[11] S. Mirjalili and S. Z. M. Hashim, "A new hybrid PSOGSA algorithm for function optimization," in *Computer and Information Application (ICCIA), 2010 International Conference on*, 2010, pp. 374-377.
[12] S. Mirjalili, S. Z. Mohd Hashim, and H. Moradian Sardroudi, "Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm," *Applied Mathematics and Computation*, vol. 218, pp. 11125-11137, 2012.
[13] J. H. Holland, "Genetic algorithms," *Scientific american*, vol. 267, pp. 66-72, 1992.
[14] D. Goldberg, "Genetic Algorithms in optimization, search and machine learning," *Addison Wesley, New York. Eiben AE, Smith JE (2003) Introduction to Evolutionary Computing. Springer. Jacq J, Roux C (1995) Registration of non-segmented images using a genetic algorithm. Lecture notes in computer science*, vol. 905, pp. 205-211, 1989.
[15] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, pp. 341-359, 1997.
[16] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *Evolutionary Computation, IEEE Transactions on*, vol. 3, pp. 82-102, 1999.
[17] D. Fogel, *Artificial intelligence through simulated evolution*: Wiley-IEEE Press, 2009.
[18] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)," *Evolutionary Computation*, vol. 11, pp. 1-18, 2003.
[19] I. Rechenberg, "Evolution strategy," *Computational Intelligence: Imitating Life*, vol. 1, 1994.
[20] J. R. Koza, "Genetic programming," 1992.
[21] D. Simon, "Biogeography-based optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 12, pp. 702-713, 2008.
[22] B. Webster and P. J. Bernhard, "A local search optimization algorithm based on natural principles of gravitation," in *Proceedings of the 2003 International Conference on Information and Knowledge Engineering (IKE'03), Las Vegas, Nevada, USA*, 2003, pp. 255-261.
[23] O. K. Erol and I. Eksin, "A new optimization method: big bang–big crunch," *Advances in Engineering Software*, vol. 37, pp. 106-111, 2006.
[24] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "GSA: a gravitational search algorithm," *Information sciences*, vol. 179, pp. 2232-2248, 2009.
[25] A. Kaveh and S. Talatahari, "A novel heuristic optimization method: charged system search," *Acta Mechanica*, vol. 213, pp. 267-289, 2010.
[26] R. A. Formato, "Central force optimization: A new metaheuristic with applications in applied electromagnetics," *Progress In Electromagnetics Research*, vol. 77, pp. 425-491, 2007.
[27] B. Alatas, "ACROA: Artificial Chemical Reaction Optimization Algorithm for global optimization," *Expert Systems with Applications*, vol. 38, pp. 13170-13180, 2011.
[28] A. Hatamlou, "Black hole: A new heuristic optimization approach for data clustering," *Information sciences*, 2012.
[29] A. Kaveh and M. Khayatazad, "A new meta-heuristic method: Ray Optimization," *Computers & Structures*, vol. 112, pp. 283-294, 2012.
[30] H. Du, X. Wu, and J. Zhuang, "Small-world optimization algorithm for function optimization," in *Advances in Natural Computation*, ed: Springer, 2006, pp. 264-273.
[31] H. Shah-Hosseini, "Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimisation," *International Journal of Computational Science and Engineering*, vol. 6, pp. 132-140, 2011.
[32] F. F. Moghaddam, R. F. Moghaddam, and M. Cheriet, "Curved Space Optimization: A Random Search based on General Relativity Theory," *arXiv preprint arXiv:1208.2214*, 2012.
[33] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature inspired cooperative strategies for optimization (NICSO 2010)*, ed: Springer, 2010, pp. 65-74.
[34] H. A. Abbass, "MBO: Marriage in honey bees optimization-A haplometrosis polygynous swarming approach," in *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, 2001, pp. 207-214.
[35] X. Li, "A new intelligent optimization-artificial fish swarm algorithm," *Doctor thesis, Zhejiang University of Zhejiang, China*, 2003.
[36] M. Roth, "Termite: A swarm intelligent routing algorithm for mobile wireless ad-hoc networks," 2005.
[37] P. C. Pinto, T. A. Runkler, and J. M. Sousa, "Wasp swarm algorithm for dynamic MAX-SAT problems," in *Adaptive and Natural Computing Algorithms*, ed: Springer, 2007, pp. 350-357.
[38] A. Mucherino and O. Seref, "Monkey search: a novel metaheuristic search for global optimization," in *AIP conference proceedings*, 2007, p. 162.
[39] X. Lu and Y. Zhou, "A novel global convergence algorithm: bee collecting pollen algorithm," in *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence*, ed: Springer, 2008, pp. 518-525.
[40] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, 2009, pp. 210-214.
[41] Y. Shiqin, J. Jianjun, and Y. Guangxing, "A Dolphin Partner Optimization," in *Intelligent Systems, 2009. GCIS'09. WRI Global Congress on*, 2009, pp. 124-128.
[42] X.-S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," *International Journal of Bio-Inspired Computation*, vol. 2, pp. 78-84, 2010.
[43] A. Askarzadeh and A. Rezazadeh, "A new heuristic optimization algorithm for modeling of proton exchange membrane fuel cell: bird mating optimizer," *International Journal of Energy Research*, 2012.
[44] A. H. Gandomi and A. H. Alavi, "Krill Herd: a new bio-inspired optimization algorithm," *Communications in Nonlinear Science and Numerical Simulation*, 2012.
[45] W.-T. Pan, "A new fruit fly optimization algorithm: taking the financial distress model as an example," *Knowledge-Based Systems*, vol. 26, pp. 69-74, 2012.
[46] L. D. Mech, "Alpha status, dominance, and division of labor in wolf packs," *Canadian Journal of Zoology*, vol. 77, pp. 1196-1203, 1999.
[47] C. Muro, R. Escobedo, L. Spector, and R. Coppinger, "Wolf-pack (Canis lupus) hunting strategies emerge from simple rules in computational simulations," *Behavioural processes*, vol. 88, pp. 192-197, 2011.
[48] J. Digalakis and K. Margaritis, "On benchmarking functions for genetic algorithms," *International journal of computer mathematics*, vol. 77, pp. 481-506, 2001.
[49] M. Molga and C. Smutnicki, "Test functions for optimization needs," *Test functions for optimization needs*, 2005.
[50] X.-S. Yang, "Test problems in optimization," *arXiv preprint arXiv:1008.0549*, 2010.

[51] S. Mirjalili and A. Lewis, "S-shaped versus V-shaped transfer functions for binary Particle Swarm Optimization," *Swarm and Evolutionary Computation*, vol. 9, pp. 1-14, 2013.
[52] J. Liang, P. Suganthan, and K. Deb, "Novel composition test functions for numerical global optimization," in *Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE*, 2005, pp. 68-75.
[53] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," *KanGAL Report*, vol. 2005005, 2005.
[54] F. van den Bergh and A. Engelbrecht, "A study of particle swarm optimization particle trajectories," *Information sciences*, vol. 176, pp. 937-971, 2006.
[55] J. S. Arora, *Introduction to optimum design*: Academic Press, 2004.
[56] A. D. Belegundu, "Study of mathematical programming methods for structural optimization," *Dissertation Abstracts International Part B: Science and Engineering[DISS. ABST. INT. PT. B- SCI. & ENG.]*, vol. 43, p. 1983, 1983.
[57] C. A. Coello Coello and E. Mezura Montes, "Constraint-handling in genetic algorithms through the use of dominance-based tournament selection," *Advanced Engineering Informatics*, vol. 16, pp. 193-203, 2002.
[58] Q. He and L. Wang, "An effective co-evolutionary particle swarm optimization for constrained engineering design problems," *Engineering Applications of Artificial Intelligence*, vol. 20, pp. 89-99, 2007.
[59] E. Mezura-Montes and C. A. C. Coello, "An empirical study about the usefulness of evolution strategies to solve constrained optimization problems," *International Journal of General Systems*, vol. 37, pp. 443-473, 2008.
[60] C. A. Coello Coello, "Use of a self-adaptive penalty approach for engineering optimization problems," *Computers in Industry*, vol. 41, pp. 113-127, 2000.
[61] M. Mahdavi, M. Fesanghary, and E. Damangir, "An improved harmony search algorithm for solving optimization problems," *Applied Mathematics and Computation*, vol. 188, pp. 1567-1579, 2007.
[62] F. Huang, L. Wang, and Q. He, "An effective co-evolutionary differential evolution for constrained optimization," *Applied Mathematics and Computation*, vol. 186, pp. 340-356, 2007.
[63] X. S. Yang, *Nature-inspired metaheuristic algorithms*: Luniver Press, 2011.
[64] A. Carlos and C. COELLO, "Constraint-handling using an evolutionary multiobjective optimization technique," *Civil Engineering Systems*, vol. 17, pp. 319-346, 2000.
[65] K. Deb, "Optimal design of a welded beam via genetic algorithms," *AIAA journal*, vol. 29, pp. 2013-2015, 1991.
[66] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer methods in applied mechanics and engineering*, vol. 186, pp. 311-338, 2000.
[67] K. S. Lee and Z. W. Geem, "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice," *Computer methods in applied mechanics and engineering*, vol. 194, pp. 3902-3933, 2005.
[68] K. Ragsdell and D. Phillips, "Optimal design of a class of welded structures using geometric programming," *ASME Journal of Engineering for Industries*, vol. 98, pp. 1021-1025, 1976.
[69] K. Deb and A. S. Gene, "A robust optimal design technique for mechanical component design," presented at the D. Dasgupta, Z. Michalewicz (Eds.), Evolutionary Algorithms in Engineering Applications, Berlin, 1997.
[70] A. Kaveh and S. Talatahari, "An improved ant colony optimization for constrained engineering design problems," *Engineering Computations: Int J for Computer-Aided Engineering*, vol. 27, pp. 155-182, 2010.
[71] B. Kannan and S. N. Kramer, "An augmented Lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design," *Journal of mechanical design*, vol. 116, p. 405, 1994.
[72] E. Sandgren, "Nonlinear integer and discrete programming in mechanical design," 1988, pp. 95-105.
[73] C. Caer, X. Le Roux, D. Marris-Morini, N. Izard, L. Vivien, D. Gao, and E. Cassan, "Dispersion engineering of wide slot photonic crystal waveguides by Bragg-like corrugation of the slot," *Photonics Technology Letters, IEEE*, vol. 23, pp. 1298-1300, 2011.
[74] T. Baba, "Slow light in photonic crystals," *Nature Photonics*, vol. 2, pp. 465-473, 2008.
[75] Y. Zhai, H. Tian, and Y. Ji, "Slow light property improvement and optical buffer capability in ring-shape-hole photonic crystal waveguide," *Lightwave Technology, Journal of*, vol. 29, pp. 3083-3090, 2011.
[76] D. Wang, J. Zhang, L. Yuan, J. Lei, S. Chen, J. Han, and S. Hou, "Slow light engineering in polyatomic photonic crystal waveguides based on square lattice," *Optics Communications*, vol. 284, pp. 5829-5832, 2011.
[77] S. M. Mirjalili and S. Mirjalili, "Light property and optical buffer performance enhancement using Particle Swarm Optimization in Oblique Ring-Shape-Hole Photonic Crystal Waveguide," in *Photonics Global Conference (PGC), 2012*, 2012, pp. 1-4.
[78] S. M. Mirjalili, K. Abedi, and S. Mirjalili, "Optical buffer performance enhancement using Particle Swarm Optimization in Ring-Shape-Hole Photonic Crystal Waveguide," *Optik - International Journal for Light and Electron Optics*, vol. 124, pp. 5989-5993, 2013.
[79] S. M. Mirjalili, S. Mirjalili, and A. Lewis, "A Novel Multi-objective Optimization Framework for Designing Photonic Crystal Waveguides," *Photonics Technology Letters, IEEE*, vol. 26, pp. 146-149, 2014.
[80] S. M. Mirjalili, S. Mirjalili, A. Lewis, and K. Abedi, "A tri-objective Particle Swarm Optimizer for designing line defect Photonic Crystal Waveguides," *Photonics and Nanostructures - Fundamentals and Applications*.
[81] J. Wu, Y. Li, C. Peng, and Z. Wang, "Wideband and low dispersion slow light in slotted photonic crystal waveguide," *Optics Communications*, vol. 283, pp. 2815-2819, 2010.