A comparison of Meta-heuristic based optimization methods using standard benchmarks*

Enol García[1], José R. Villar[1], Camelia Chira[2], and Javier Sedano[3]

[1] University of Oviedo, Computer Science Department, Oviedo, Spain
{garciaenol,villarjose}@uniovi.es
[2] University of Babes Boliay, Department of Computer Science, Cluj-Napocav, Romania
camelia.chira@ubbcluj.ro
[3] Instituto Tecnológico de Castilla y León, Burgos, Spain
javier.sedano@itcl.es

Abstract. Optimization problems are a type of problem in which multiple solutions satisfy the problem's constraints, so not only must a good solution be found, but the objective is to find the best solution among all those considered valid. Optimization problems can be solved by using deterministic and stochastic algorithms. Those categories can be divided into different kinds of problems. One of the categories inside stochastic algorithms is metaheuristics. This work implements three well-known meta-heuristics –Grey Wolf Optimizer, Whale Optimization Algorithm, and Moth Flame Optimizer–, and compares them using ten mathematical optimization problems that combine non-constrained from other studies and constrained problems from CEC2017 competition. Results show the Grey Wolf Optimizer as the method with faster convergence and best fitness for almost all the problems. This work aims to implement and compare various metaheuristics to carry out future work on solving various real-world problems.

Keywords: Meta-heuristics · Optimization · Benchmarking

* This research has been founded by European Union's Horizon 2020 research and innovation programme (project DIH4CPS) under the Grant Agreement no 872548. Furthermore, this research has been funded by the SUDOE Interreg Program -grant INUNDATIO-, by the Spanish Ministry of Economics and Industry, grant PID2020-112726RB-I00, by the Spanish Research Agency (AEI, Spain) under grant agreement RED2018-102312-T (IA-Biomed), by CDTI (Centro para el Desarrollo Tecnológico Industrial) under projects CER-20211003 and CER-20211022 , by and Missions Science and Innovation project MIG-20211008 (INMERBOT). Also, by Principado de Asturias, grant SV-PA-21-AYUD/2021/50994 and by ICE (Junta de Castilla y León) under project CCTT3/20/BU/0002.

1 Introduction

Optimization problems are a type of problem in which multiple solutions satisfy the problem's constraints, so not only must a valid solution be found, but the objective is to find the best solution among all those considered valid. The algorithms that propose to solve this optimization problem are usually grouped into two main categories: i) deterministic and ii) stochastic. On deterministic algorithms, the results are only determined by the input data. So, if the algorithm is run twice with the same input data, it will return the same results. An example of this deterministic algorithm could be the graph search[18]. Stochastic algorithms incorporate an element of randomness, meaning that two runs with the same program and input data do not produce the same execution. In stochastic algorithms, this touch of randomness is used to limit the solutions or guide the search for solutions to where the best solution is believed to be instead of traversing the entire space of valid solutions.

Stochastic algorithms can be divided into two categories: heuristic and meta-heuristic. Heuristic algorithms are given additional information about the problem to guide their search or learn during the search by trial and error. An example of a heuristic algorithm can be A*[6]. In the case of meta-heuristics, a more general strategy is chosen: they try to learn the search space to make the search for the optimal solution more efficient, which means that they can be applied to different types of problems in a more straightforward way. To learn from the problem, meta-heuristics often imitate existing behaviors: music[5], sports[4], mathematics[12], physics[9, 7], chemistry[8], biology[3, 10, 14, 13, 11], societies[15].

This work aims to implement and compare the performance of some meta-heuristics. Different works can be found in the literature doing this kind of comparison in two ways: using a benchmark with generic mathematical problems[16, 2] or applying the meta-heuristic against concrete problems[1, 17]. This paper will compare the different meta-heuristics using a benchmark of mathematical problems.

The organization of this paper is as follows. Firstly, a description of the meta-heuristics implemented and the problems used for optimization is in section 2. Then, the experimental results are discussed in section 3. Finally, the conclusions of this paper are in section 4.

2 Materials and Methods

This section outlines each of the three metaheuristic-based optimization methods –Sections 2.1, 2.2 and 2.3 for the Grey Wolf Optimizer (GWO), the Whale Optimization Algorithm (WOA) and Moth Flame Optimization (MFO), correspondingly. Furthermore, the section ends with the description of the standardized problems for this comparison (see Sect. 2.4) followed by the experimental setup (see Sect. 2.5).

2.1 Grey Wolf Optimizer (GWO)

The GWO meta-heuristic was proposed in [14]; it mimics the hunting behavior of grey wolves. This meta-heuristic employs a series of agents that search for solutions for its operation. These agents in charge of searching for the best solution are the wolves. As in a wolf pack, the solution-seeking agents have different social statuses. The best wolves in the pack are the alpha wolves, followed by the beta, delta, and omega wolves.

Within the pack, the wolves with the highest status –the alpha role–, are in charge of leading the pack and making decisions: when to hunt, when to sleep, where the pack migrates to, and others. The beta role –the second-highest status–, help the alphas in their decision-making and lead the lower-level wolves. The delta role represents the third and lowest social level. The deltas function as a scapegoat within the social structure. They are the last echelon that can give orders. They are in charge of transmitting the orders and directives of the higher echelons to the next level. The lowest echelon is occupied by the omega, who is in charge of carrying out the work within the herd: hunting, fighting and exploring.

The mathematical behavior of meta-heuristics will mimic the hunting process of wolves. The best agent searching for prey will be the alpha, the second the beta, the third the delta, and the rest will be considered omegas. The position and the information of the superior individuals –alpha, beta, and delta roles–, determine the movements that the omegas have to do. The omegas move through the solution space to find new prey or solutions. The meta-heuristic process is iterative. In each iteration, the position of the hunters will be modified to find better prey.

Algorithm 1 Algorithm of the Grey Wolf Optimizer (GWO) metaheuristic

```
Initialize the grey wolf population X_i(i = 1, 2, ...n))
Initialize iteration parameters
Calculate the fitness of each search agent
X_α = the best search agent
X_β =the second best search agent
X_δ =the third best search agents
while t < Max number of iterations do
    for each search agent do
        Update the position of the current search agent
    end for
    Update iteration parameters
    Calculate the fitness of each search agent
    Update X_α, X_β and X_δ
    t ← t + 1
end while
```

2.2 Whale Optimization Algorithm (WOA)

Whale Optimization Algorithm (WOA) is a meta-heuristic proposed in [13] that mimics the behavior of whales in their search for food. It is based on a series of agents–whales–which move through the solution space searching for food.

The metaheuristic proposes three behaviors for the whale: i) search for prey, ii) envelop the prey, and iii) attack. At all times, a random value determines the behavior of the whales. With a probability of 50%, the method will be considered to be in the phase of attacking the prey. The remaining 50% will determine that we are in one of the other states. Another value will be used to determine whether the prey is searched for or encircled. This second value will be calculated based on our iteration and a random value. Thus, at the beginning of the execution, it is more likely to select the state of searching for a dam, while as the iterations evolve, it will be more likely to go around the dam. These two phases correspond to the exploration and exploitation of the solution space.

Algorithm 2 Algorithm of the Whale Optimization Algorithm (WOA) meta-heuristic

Initialize the whales population $X_i (i = 1, 2, ..., n)$
Calculate the fitness of each search agent
X^* = the best search agent
while t < Max number of iterations **do**
  for each search agent **do**
    Update iteration parameters
    $p \leftarrow random \in [0, 1]$
    $r \leftarrow random \in [0, 1]$
    $A \leftarrow 2 * (2 - t * 2/Max\_iter) * r - (2 - t * 2/Max\_iter)$
    if p<0.5 **then**
      if |A| < 1 **then**
        Update the position of the current search agent by an spiral movement with a radius depending on the current iteration (Attacking)
      else
        Select a random search agent (X_{rand})
        Update the position of the current search agent by moving him on the direction of X_{rand} (Searching for a prey)
      end if
    else
      Update the position of the current search agent by moving him using a small movement (Encircling the prey)
    end if
  end for
  Check if any search agent goes beyond the search space and amend it
  Calculate the fitness of each search agent
  Update X^* if there is a better solution
  $t \leftarrow t + 1$
end while

2.3 Moth Flame Optimizer (MFO)

Moth Flame Optimizer (MFO)[11] is a meta-heuristic that mimics the natural behavior of moths when they fly towards the light of a flame. In this meta-heuristic, the moths are the agents that search for the best solutions, which are candle flames. By observing the behavior of the moths, it is observed how they fly towards the flames in a spiral shape.

To implement this metaheuristic, we will start with a random population of moths evenly distributed in space. It is assumed that there is a flame at the position of the moths, i.e., a solution. The metaheuristic consists of an iterative process. A list of the best flames - best solutions - is used in each iteration. Each moth will select a flame and perform a spiral movement towards it in each iteration. During the spiral movement, it is possible to find other better flames, in which case, the list of best flames will change, and the moth will select a new flame.

Algorithm 3 Algorithm of the Moth Flame Optimizer (MFO) metaheuristic

```
Initialize the moth population X_i(i = 1, 2, ..., n)
Calculate the fitness of each search agent
Initialize the best flames population. Initially same as moth population
while t < Max number of iterations do
    Update iteration parameters
    for each moth on the population do
        Assign a flame
        Update position of the moth moving to the assigned flame
    end for
    t ← t + 1
end while
```

2.4 Standardized problems

The experimentation phase uses two different types of problems: unconstrained and constrained. Non-Constrained Optimization problems only use a function $f(x) \in \mathbb{R}$; the goal is to search the value of $x \in \mathbb{R}$ that minimizes the value of $f(x)$. Up to 5 non-constrained problems -extracted from [14]- are used in the comparison; these functions are listed in Table 1.

Besides, a unconstrained function $f(x) \in \mathbb{R}, x \in \mathbb{R}$, includes one or more restrictions and conditions the either the function, the x value or a third function $g(x) \in \mathbb{R}$ must satisfy. For this study, the functions included in Table 2, extracted from the CEC2017 competition [19], are used in this comparison.

2.5 Experimental setup

Each problem is solved using the three metaheuristics: GWO, WOA, and MFO. Each optimization problem evolved during 1000 iterations. Besides, the meth-

ID	Function
f01	$f(X) = \sum_{i=1}^{n} x_i^2, X \in [-100, 100]$
f02	$f(X) = \sum_{i=1}^{n} \lvert x \rvert + \prod_{i=1}^{n} \lvert x \rvert, X \in [-10, 10]$
f03	$f(X) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2, X \in [-100, 100]$
f08	$f(X) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert}), X \in [-500, 500]$
f09	$f(X) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10], X \in [-5.12, 5.12]$

Table 1: List of unconstrained functions used in this research.

ods' parameters were chosen among the best reported in the literature. Each method is run 100 times for each problem to obtain trustworthy metrics about its performance. The following metrics are proposed to measure the performance of the meta-heuristic optimization algorithms:

- Execution time
- Fitness of the best solution found
- Average fitness of the solutions present in the last population

3 Results and discussion

Results include both time consumption and fitness values. Table 3 shows the mean and standard deviation of the execution time of each of the algorithms. As can be seen, GWO is up to 5 times faster than the rest for all types of problems, while WOA and MFO have very similar execution times to each other.

On the other hand, Table 4 shows the average of the best fitness found in each algorithm run. For all of the problems, it can be observed how MFO always offers a worse fitness than the other two meta-heuristics. Even for MFO, it is observed that it cannot find a feasible solution in the case of problem C19. Analyzing GWO and WOA, it is seen that both can reach the optimal solution with unconstrained problems. Both meta-heuristics obtain a solution with very similar fitness to the constrained problems.

Besides, Table 5 shows the average fitness of the solutions of the last population of each run. It can be seen that these data are almost identical to those in Table 4, i.e., all meta-heuristics have already stabilized on a solution and are not exploring better solutions.

As seen from the tables, the general tendency is for the GWO meta-heuristic to be the fastest in delivering results, while MFO always takes the longest to run. As for the fitness values, GWO offers very similar values to WOA, but MFO always has a higher value. Figure 1 shows this fact for problem f09, depicting the comparison in times and fitness for all of the runs. There is no doubt that GWO

ID	Function & Constrains
C01	$f(X) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$
	$g(X) = \sum_{i=1}^{n} [x_i^2 - 5000\cos(0.1\pi x) - 4000] \leq 0$
	$x \in [-100, 100]$
C04	$f(X) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$
	$g_1(X) = \sum_{i=1}^{n} x\sin(2x) \leq 0$
	$g_2(X) = \sum_{i=1}^{n} x\sin(x) \leq 0$
	$x \in [-10, 10]$
C05	$f(X) = \sum_{i=1}^{n-1} [100(x_i^2 - x_{i+1})^2 + (x_1 - 1)^2]$
	$g_1(X) = \sum_{i=1}^{n} [y_i^2 - 50\cos(2\pi y_i) - 40] \leq 0, y_i = M_1 * X$
	$g_2(X) = \sum_{i=1}^{n} [w_i^2 - 50\cos(2\pi w_i) - 40] \leq 0, w_i = M_2 * X$
	$x \in [-10, 10]$
C13	$f(X) = \sum_{i=1}^{n-1} [100(x_i^2 - x_{i+1})^2 + (x_1 - 1)^2]$
	$g_1(X) = \sum_{i=1}^{n} [y_i^2 - 10\cos(2\pi y_i) + 10] \leq 0$
	$g_2(X) = \sum_{i=1}^{n} (x_i - 60) \leq 0$
	$g_3(X) = \sum_{i=1}^{n} x_i \leq 0$
	$x \in [-100, 100]$
C19	$f(X) = \sum_{i=1}^{n} (\lvert x_i\rvert^{0.5} + 2\sin x_i^3)$
	$g_1(X) = \sum_{i=1}^{n-1} (-10\exp(-0.2\sqrt{x_i^2 + x_{i+1}^2})) \leq 0$
	$g_2(X) = \sum_{i=1}^{n} 2x_i - 0.5 \leq 0$
	$x \in [-50, 50]$

Table 2: List of contrained functions used in this research.

Method	f01	f02	f03	f08	f09
GWO	**44.10**	**44.00**	**74.83**	**48.60**	**51.80**
	(0.97)	(1.02)	(0.82)	(1.06)	(1.20)
WOA	258.97	247.73	365.45	261.45	270.82
	(4.22)	(4.93)	(6.34)	(4.78)	(4.05)
MFO	252.71	250.16	371.17	262.84	273.34
	(4.56)	(4.36)	(4.64)	(4.54)	(4.74)
Method	**C01**	**C04**	**C05**	**C13**	**C19**
GWO	**86.40**	**56.83**	**74.62**	**62.72**	**68.35**
	(1.71)	(0.97)	(1.41)	(0.85)	(1.40)
WOA	409.90	303.22	431.47	368.10	341.28
	(6.18)	(4.74)	(6.36)	(5.78)	(9.26)
MFO	421.56	313.06	432.06	387.80	308.13
	(5.92)	(5.22)	(5.82)	(4.86)	(6.69)

Table 3: Mean and standard deviation –just under the mean value and delimited with parenthesis– of the execution time of each meta-heuristics solving the different unconstrained problems. The best results are marked in bold letters.

Method	f01	f02	f03	f08	f09
GWO	**0.00**	**0.00**	**0.00**	-743.49	**0.00**
	(0.00)	(0.00)	(0.00)	(129.67)	(0.00)
WOA	**0.00**	**0.00**	**0.00**	**-7313.57**	**0.00**
	(0.00)	(0.00)	(0.00)	(2127.27)	(0.00)
MFO	680.56	0.89	3.95e5	-6510.48	78.57
	(758.83)	(1.94)	(999.61)	(1899.98)	(30.76)
Method	**C01**	**C04**	**C05**	**C13**	**C19**
GWO	**0.00**	-43.69	27.21	**70.46**	**-0.75**
	(0.00)	(5.36)	(0.00)	(45.64)	(0.34)
WOA	**0.00**	-235.90	**26.15**	90.89	-0.14
	(0.00)	(76.72)	(0.00)	(186.75)	(0.32)
MFO	3.97e5	**-300**	3.49e5	1.13e9	inf height
	(8.94e4)	(0.00)	(5.76e5)	(1.50e9)	

Table 4: Mean and standard deviation –just below the mean value and delimited with parenthesis–, of the best fitness of each meta-heuristic solving each problem. The best values are remarked in bold letters.

Method	**f01**	**f02**	**f03**	**f08**	**f09**
GWO	**0.00**	**0.00**	**0.00**	-743.49	**0.00**
	(0.00)	(0.00)	(0.00)	(129.67)	(0.00)
WOA	**0.00**	**0.00**	**0.00**	**-7313.37**	**0.00**
	(0.00)	(0.00)	(0.00)	(2127.27)	(0.00)
MFO	680.56	0.89	3.95e5	-6510.48	78.57
	(758.83)	(1.94)	(999.61)	(1899.98)	(30.76)
Method	**C01**	**C04**	**C05**	**C13**	**C19**
GWO	**0.00**	-43.69	27.21	**70.46**	**-0.75**
	(0.00)	(5.36)	(0.00)	(45.64)	(0.34)
WOA	**0.00**	-235.88	**26.15**	90.90	-0.14
	(0.00)	(76.71)	(0.80)	(186.78)	(0.32)
MFO	3.97e5	**-300**	3.49e5	1.13e9	inf height
	(8.94e4)	(0.00)	(5.76e5)	(1.50e9)	

Table 5: Mean and standard deviation –just below the mean value and delimited with parenthesis–, of the mean fitness of each meta-heuristic solving each problem. The best values are remarked with bold letters

outperforms the other two methods if we consider the two criteria simultaneously.

4 Conclusion

In this study, three known meta-heuristics–Grey Wolf Optimizer, Whale Optimization Algorithm, and Moth Flame Optimizer–were implemented and tested against ten optimization problems, five of which were unconstrained and five of which were constrained. In the analysis of the results, it was observed that the meta-heuristic that offered the fastest results was Grey Wolf Optimizer, but this is not the only criterion to be analyzed. In addition to the execution time, we also examined the minimum value found for optimizing the function. In this case, the values of the Grey Wolf Optimizer were very close to those of the Whale Optimization Algorithm. The meta-heuristic that yielded the worst data for the minimum value of the function was Moth Flame Optimizer, which, even for some more complicated problems, could not find any solution that satisfies the constraints of the function.

In this work, we have tried to collect the most common metaheuristics found in the literature, implement them and conduct a comparative study between them to analyze their performance. This work aims to be the starting point of a line of research in which we will try to solve other problems with real-world applications.

Future work in this line of research will adapt the implementations made in this study to solve problems such as multi-robot path planning, time slot allocation, or collision-free multi-robot trajectory determination. Future work also aims to propose some improvements to improve the performance of these algorithms.

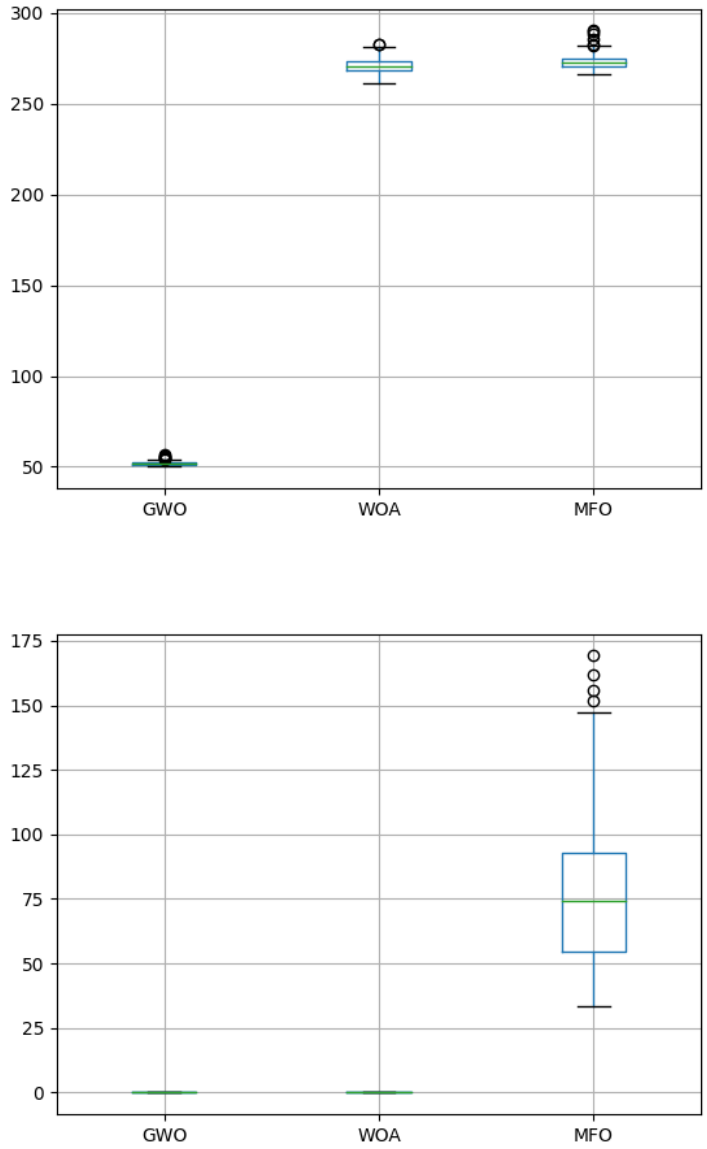

Fig. 1: Comparison box-plots for metrics for function f09. The upper part depicts the time consumption, while the bottom part shows the fitness values.

References

1. Azimi, Z.N.: Comparison of metaheuristic algorithms for examination timetabling problem. Journal of Applied Mathematics and Computing **16**(1), 337 (Mar 2004). https://doi.org/10.1007/BF02936173
2. Bloomfield, M.W., Herencia, J.E., Weaver, P.M.: Analysis and benchmarking of meta-heuristic techniques for lay-up optimization. Computers and Structures **88**(5), 272–282 (2010). https://doi.org/10.1016/j.compstruc.2009.10.007
3. Dorigo, M., Di Caro, G.: Ant colony optimization: a new meta-heuristic. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406). vol. 2, pp. 1470–1477 Vol. 2 (1999). https://doi.org/10.1109/CEC.1999.782657
4. Fadakar, E., Ebrahimi, M.: A new metaheuristic football game inspired algorithm. In: 2016 1st Conference on Swarm Intelligence and Evolutionary Computation (CSIEC). pp. 6–11 (2016). https://doi.org/10.1109/CSIEC.2016.7482120
5. Geem, Z.W., Kim, J.H., Loganathan, G.: A new heuristic optimization algorithm: Harmony search. SIMULATION **76**(2), 60–68 (2001). https://doi.org/10.1177/003754970107600201
6. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics **4**(2), 100–107 (1968). https://doi.org/10.1109/TSSC.1968.300136
7. Hatamlou, A.: Black hole: A new heuristic optimization approach for data clustering. Information Sciences **222**, 175–184 (2013). https://doi.org/10.1016/j.ins.2012.08.023, including Special Section on New Trends in Ambient Intelligence and Bio-inspired Systems
8. Irizarry, R.: A generalized framework for solving dynamic optimization problems using the artificial chemical process paradigm: Applications to particulate processes and discrete dynamic systems. Chemical Engineering Science **60**(21), 5663–5681 (2005). https://doi.org/10.1016/j.ces.2005.05.028
9. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983). https://doi.org/10.1126/science.220.4598.671
10. Mirjalili, S.: The ant lion optimizer. Advances in Engineering Software **83**, 80–98 (2015). https://doi.org/10.1016/j.advengsoft.2015.01.010
11. Mirjalili, S.: Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowledge-Based Systems **89**, 228–249 (2015). https://doi.org/10.1016/j.knosys.2015.07.006
12. Mirjalili, S.: Sca: A sine cosine algorithm for solving optimization problems. Knowledge-Based Systems **96**, 120–133 (2016). https://doi.org/10.1016/j.knosys.2015.12.022
13. Mirjalili, S., Lewis, A.: The whale optimization algorithm. Advances in Engineering Software **95**, 51–67 (2016). https://doi.org/10.1016/j.advengsoft.2016.01.008
14. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Advances in Engineering Software **69**, 46–61 (2014). https://doi.org/10.1016/j.advengsoft.2013.12.007
15. Mousavirad, S.J., Ebrahimpour-Komleh, H.: Human mental search: a new population-based metaheuristic optimization algorithm. Applied Intelligence **47**(3), 850–887 (Oct 2017). https://doi.org/10.1007/s10489-017-0903-6
16. Parejo, J.A., Ruiz-Cortés, A., Lozano, S., Fernandez, P.: Metaheuristic optimization frameworks: a survey and benchmarking. Soft Computing **16**(3), 527–561 (Mar 2012). https://doi.org/10.1007/s00500-011-0754-8

17. Sonmez, M.: Performance comparison of metaheuristic algorithms for the optimal design of space trusses. Arabian Journal for Science and Engineering **43**(10), 5265–5281 (Oct 2018). https://doi.org/10.1007/s13369-018-3080-y
18. Williams, M.L., Wilson, R.C., Hancock, E.R.: Deterministic search for relational graph matching. Pattern Recognition **32**(7), 1255–1271 (1999). https://doi.org/10.1016/S0031-3203(98)00152-6
19. Wu, G., Mallipeddi, R., Suganthan, P.: Problem definitions and evaluation criteria for the cec 2017 competition and special session on constrained single objective real-parameter optimization. Tech. rep., IEEE Congress on Evolutionary Computation (10 2016)