Improved Discrete Grey Wolf Optimizer

Benoit Martin
IntuiSense Technologies
Gemenos, France
benoit.martin@intui-sense.com

Julien Marot
Aix Marseille Univ., CNRS
Institut Fresnel
Marseille, France
julien.marot@fresnel.fr

Salah Bourennane
Centrale Marseille, CNRS
Institut Fresnel
Marseille, France
salah.bourennane@fresnel.fr

***Abstract*—Grey wolf optimizer (GWO) is a bio-inspired iterative optimization algorithm which simulates the hunting process of a wolf pack guided by three leaders. In this paper, a novel discrete GWO is proposed: a random leader selection is performed, and the probability for the main leader to be selected increases at the detriment of the other leaders across iterations. The proposed discrete GWO is compared to another discrete version of GWO, using standard test functions.**

***Index Terms*—bio-inspired optimization, discrete space, grey wolf**

I. Introduction

Bio-inspired optimization methods mimic the behavior of animals in nature to maximize or minimize a real function by systematically choosing input values from within an allowed search space and computing the value of the function. With the emergence of graphs and discrete signal processing (see [1], [2] and references inside), it is more and more relevant to innovate on bio-inspired optimization methods which are specifically dedicated to discrete search spaces.

Relation to prior work in the field:

The first bio-inspired optimization method is particle swarm optimization (PSO) [3]. Afterwards, grey wolf optimization (GWO) was proposed in [4]. The GWO algorithm mimics the leadership hierarchy and hunting mechanism of grey wolves to create update rules for the search agents. Four types of grey wolves, the alpha, beta, delta, and the omega are employed for simulating the leadership hierarchy. A number of variants are also proposed to improve the performance of basic GWO that include a hybrid version of GWO with PSO [5], and a binary GWO [6] solves a combinatorial problem for classification purposes. The very last versions of GWO aim at solving discrete problems, such as the multiobjective discrete GWO (MODGWO) whose implementation is detailed in [7], [8].

Main contributions:

The purpose of our paper is to develop an improved discrete grey wolf optimizer: update rules are proposed for wolves which permit to distinguish clearly between an exploration phase, and an exploitation phase, respecting thereby the primal philosophy of the grey wolf optimizer, and taking advantage of its properties.

Outline:

In section II we review the computational aspects of grey wolf optimization; in section III, we present the proposed improved discrete GWO. In section IV, a comparative performance evaluation is performed on several benchmark functions.

Notations:

Scalars are denoted by italic lowercase or uppercase roman, like a or A; vectors by boldface lowercase roman, like $\mathbf{a}$; matrices by boldface uppercase roman, like $\mathbf{A}$; Manifolds are denoted by blackboard bold like $\mathbb{A}$.

II. Computational background

Optimization algorithms aim at estimating the best values of N parameters $K_1, K_2, \ldots, K_N$, where $N \geq 1$. In this section we focus on the computational aspects of GWO and MODGWO. The following notations will be used:

- N is the number of expected parameters, which are indexed with i.

ISBN 978-90-827970-1-5 © EURASIP 2018

• iter denotes one iteration and $\mathrm{T}_{\max}$ the maximum allowed number of iterations.
• $f(\cdot)$ is the function which is minimized, also called criterion. It depends on the N parameters which should be estimated.
• $\mathbf{x}_q^{\mathbf{k}}(\mathrm{iter})$ is a vector corresponding to one search agent $q = 1, \ldots, Q$, at iteration iter. It takes a form of a vector with an N-uplet of tested values $\mathbf{x}_q^{\mathbf{k}}(\mathrm{iter}) = [K_1, K_2, \ldots, K_N]^T$.

A. Continuous Grey wolf optimization

The seminal Grey wolf optimizer (GWO) mimics the behavior of a herd of wolves [4] to search a continuous space. Four types of grey wolves are employed to simulate the leadership hierarchy: three leaders α, β, δ, and the ω wolves. Searching and attacking prey are implemented. The updated position is calculated as:

$$\mathbf{x}_q^{\mathbf{k}}(\mathrm{iter}+1) = \frac{1}{3}(\mathbf{y}^{\alpha} + \mathbf{y}^{\beta} + \mathbf{y}^{\delta}) \tag{1}$$

It results from the equal contribution of the α, the β, and the δ wolves. These contributions are computed as follows, for instance for the α:

$$\mathbf{y}^{\alpha} = \mathbf{x}_{\alpha}^{\mathbf{k}} - \mathbf{b} \cdot \mathbf{d}_{\alpha} \tag{2}$$

with: $\mathbf{d}_{\alpha} = |\mathbf{c} \cdot \mathbf{x}_{\alpha}^{\mathbf{k}} - \mathbf{x}_q^{\mathbf{k}}(\mathrm{iter})|$
The vectors $\mathbf{b}$ and $\mathbf{c}$ are calculated as $\mathbf{b} = 2\mathbf{a} \cdot \mathbf{r}_1 - \mathbf{a}$ and $\mathbf{c} = 2 \cdot \mathbf{r}_2$. In these expressions, the components of vector $\mathbf{a}$ are all equal to a, a scalar value which is a key parameter in the algorithm. The value of a decreases from 2 to 0 across the iterations. Vectors $\mathbf{r}_1$, $\mathbf{r}_2$ have random components between 0 and 1.
During the hunt, the wolves firstly diverge from each other to search for prey. Secondly, they converge to attack prey. This is mathematically modeled through the deterministic vector $\mathbf{a}$. When $a > 1$, the search agents are obliged to diverge from the prey: this is the exploration phase. Conversely, when $a \leq 1$, the search agents are obliged to attack towards the prey: this is the exploitation phase. We notice that a key parameter to balance the exploration and the exploitation phases is the deterministic vector $\mathbf{a}$. In the seminal version of GWO [4], a is regularly decreased from 2 to 0: $a = 2(1 - \frac{\mathrm{iter}}{T_{max}})$, where iter is the iteration index, and T_{max} is the maximum number of iterations. The exploration step lasts until $a = 1$, the exploitation step lasts from $a = 1$ to $a = 0$. In [9], a modified GWO (mGWO) is implemented with an exponential model for a:

$$a = 2(1 - \frac{\mathrm{iter}^2}{{\mathrm{T}_{\max}}^2}) \tag{3}$$

This permits to emphasize the exploration phase, that is, to encourage a global search, at the expense of the exploitation phase.

B. Discrete Grey wolf optimization

In [7] a multiobjective discrete version of GWO (MODGWO) is proposed, with two types of update rules:
Let $rand$ be a random real number between 0 and 1.

$$K_1 = \begin{cases} K_1^{\alpha} & \text{if} \quad rand \leq \frac{1}{3} \\ K_1^{\beta} & \text{if} \quad rand > \frac{1}{3} \quad \text{and} \quad rand \leq \frac{2}{3} \\ K_1^{\delta} & \text{if} \quad rand > \frac{2}{3} \end{cases} \tag{4}$$

The second update rule is as follows: Let $rand$ be a random real number between 0 and 1.

$$K_2 = \begin{cases} K_2^{\alpha},\ K_2^{\beta},\ \text{or}\ K_2^{\delta} & \text{if} \quad rand \leq a \\ K_2^{'} & \text{if} \quad rand > a \end{cases} \tag{5}$$

In Eq. (5), $a = 1 - \frac{\mathrm{iter}}{\mathrm{T}_{\max}}$, and $K_2^{'}$ is the second component of any wolf in the herd, taken at random. We notice that, in this algorithm,

These update rules are certainly valid for the application considered in [7]. However, we expect from the improved discrete GWO proposed in the following to obtain better results, as it better respects the philosophy of the classic GWO.

III. IMPROVED DISCRETE GWO

The idea behind the discrete grey wolf optimizer is that the search spaces are discrete, and that not only sets of values are searched but also sets of indexes. A search agent or 'wolf' at iteration iter is denoted by $\mathbf{x}_q^{\mathbf{k}}(\mathrm{iter})$ and contains N components which are possible values of expected parameters:
• For each parameter i, the number of possible values is denoted by the scalar H_i.
• Vector $\mathbf{d}_i^{val} = \left[K_i^1, \ldots, K_i^{h_i}, \ldots, K_i^{H_i}\right]^T$ is the 'search domain' for K_i, that is, it contains all

ISBN 978-90-827970-1-5 © EURASIP 2018

acceptable values for parameter K_i.
• Vector $\mathbf{d}_i^{ind} = [1, \ldots, h_i, \ldots, H_i]^T$ contains the index of each acceptable value for the i^{th} parameter. $\mathbf{d}_i^{ind}$ contains H_i integer values in increasing order.
• A wolf denoted by $\mathbf{x}_q^{\mathbf{k}}(\text{iter})$, is a vector with N components that we call vector of 'values'. These values are denoted as follows: $\mathbf{x}_q^{\mathbf{k}}(\text{iter}) = [x_1(\text{iter}), \ldots, x_i(\text{iter}), \ldots, x_N(\text{iter})]^T$, where the index q is omitted for sake of clarity.
• Vector $\mathbf{h}_q(\text{iter})$ is the vector of N indexes associated with wolf $\mathbf{x}_q^{\mathbf{k}}(\text{iter})$. These values are denoted as follows: $\mathbf{h}_q(\text{iter}) = [h_1(\text{iter}), \ldots, h_i(\text{iter}), \ldots, h_N(\text{iter})]^T$, where the index q is omitted for sake of clarity.
• Vectors $\mathbf{x}_\alpha^{\mathbf{k}}$, $\mathbf{x}_\beta^{\mathbf{k}}$, $\mathbf{x}_\delta^{\mathbf{k}}$, $\mathbf{x}_{\rho1}^{\mathbf{k}}$, $\mathbf{x}_{\rho2}^{\mathbf{k}}$ are the vectors of values for leaders α, β, δ, and for two wolves $\rho1$ and $\rho2$, selected at random among the Q wolves of the population.
• Vectors $\mathbf{h}_\alpha$, $\mathbf{h}_\beta$, $\mathbf{h}_\delta$, $\mathbf{h}_{\rho1}$, $\mathbf{h}_{\rho2}$ are the vectors of indexes for leaders α, β, δ, and for the two random wolves $\rho1$ and $\rho2$.
• The current leader in our iterative algorithm is denoted by l. Its vector of values is $\mathbf{x}_\mathrm{l}^{\mathbf{k}} = \left[x_1^\mathrm{l}, \ldots, x_i^\mathrm{l}, \ldots, x_N^\mathrm{l}\right]^T$ and its vector of indexes is $\mathbf{h}_\mathrm{l} = \left[h_1^\mathrm{l}, \ldots, h_i^\mathrm{l}, \ldots, h_N^\mathrm{l}\right]^T$.
• The scalar x_i^l is the i^{th} component of $\mathbf{x}_\mathrm{l}^{\mathbf{k}}$, and the scalar h_i^l denotes the i^{th} component of $\mathbf{h}_\mathrm{l}$.
We notice that a value located at the component with index h_i in $\mathbf{d}_i^{ind}$ is denoted by $K_i^{h_i}$. For instance, for the vector of values $\mathbf{x}_q^{\mathbf{k}}(\text{iter}) = \left[K_1^2, \ldots, K_N^5\right]^T$, the associated vector of indexes is $\mathbf{h}_q(\text{iter}) = [2, \ldots, 5]^T$.
Algorithm 1 (see below) details the proposed improved discrete GWO. It involves a parameter a, as defined in Eq. (3).

Step 5 includes the selection of a leader $\mathbf{x}_\mathrm{l}^{\mathbf{k}}$ and the update process of the index vector $\mathbf{h}_q(\text{iter})$.
As the parameter a is decreasing from 2 to 0 across the iterations, it is more and more probable for α to be chosen as the leader. The random wolves may be selected during the first part of the process, when $a > 1$, and cannot be selected during the second part of the process, when $a \leq 1$.
In the update equations at step 5b, $sgn(\cdot)$ denotes the sign function, and mod denotes the 'Modulo' operator. Δ is computed as follows:

$$\Delta = \begin{cases} 1 & \text{if} \quad \phi \leq \frac{a}{6} \\ 2 & \text{if} \quad \phi > \frac{a}{6} \quad \text{and} \quad \phi \leq \frac{2a}{6} \\ 4 & \text{if} \quad \phi > \frac{2a}{6} \quad \text{and} \quad \phi \leq \frac{3a}{6} \\ 1 & \text{if} \quad \phi > \frac{3a}{6} \end{cases} \tag{6}$$

where ϕ is a random value in $\mathbb{R}$, between 0 and 1. Across the iterations, it is more and more probable for the value 1 to be chosen, and less and less probable for the larger values such as 2 and 4 to be chosen. This is coherent with the paradigm of the original Grey Wolf Optimizer [4], where exploration is emphasized when $a > 1$ at the beginning of the process, and exploitation is emphasized when $a \leq 1$ at the end of the process. The proposed algorithm has a complexity of $O(T_{max}NQ)$.

IV. Performance evaluation

In this section, we evaluate the performances of our improved discrete GWO and comparative MODGWO [7]. When MODGWO is run, the first parameter is updated using Eq. (4), while the other parameters are updated using Eq. (5). Unimodal functions are those which exhibit only one global minimum and no relative minima, whereas multimodal functions exhibit several relative minima. Tests have been run on the two unimodal function (F_1 and F_2), and two multimodal functions (F_{16} and F_{18}) [4], [9]:

$$F_1(x) = \sum_{i=1}^{n} x_i^2, \text{ with } n = 3$$

$$F_2(x) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i|, \text{ with } n = 3$$

$$F_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + x_2^4$$

$$F_{18}(x) = \left[1 + (x_1 + x_2 + 1)^2 \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right] \left[30 + (2x_1 - 3x_2)^2 \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right]$$

Table I presents, for each function, the expected minimum value, and the search space for our improved discrete GWO and MODGWO. The results are computed over $M = 30$ independents runs. To compute the performance metrics, we consider, for the m^{th} run, $f(\mathbf{x}_\alpha^{\mathbf{k}})_m$, the fitness value obtained at

ISBN 978-90-827970-1-5 © EURASIP 2018

Algorithm 1 Pseudo-code: Improved Discrete Grey Wolf Optimization for multiple parameter estimation

Inputs: fitness function f, number N of expected parameters, small factor ϵ set by the user, to stop the algorithm, maximum number of iterations T_{max}.
For each parameter indexed by $i = 1, \dots, N$: the search space $\mathbf{d}_i^{ind}$ with H_i possible values.

1) Set iteration number $\text{iter} = 1$,
   create an initial set of index vectors $\mathbf{h}_q(\text{iter})$, $q = 1, \dots, Q$. For each index q, each component h_i^{iter} is an integer value between 1 and H_i.
   Create an initial herd composed of Q wolves $\mathbf{x}_q^{\mathbf{k}}(\text{iter})$, $q = 1, \dots, Q$ with the N required parameter values. This initial population takes the form of a matrix with Q rows and N columns. For each index q, $x_i(\text{iter}) = \mathbf{d}_i^{ind}(h_i(\text{iter}))$.
2) Evaluate fitness function value $f(\mathbf{x}_q^{\mathbf{k}}(\text{iter}))$ of each wolf $\mathbf{x}_q^{\mathbf{k}}(\text{iter})$, $q = 1, \dots, Q$.
3) Sort the wolves through their fitness value and update the wolves which hold the first, second and third best fitness value: store the corresponding vectors of indexes $\mathbf{h}_\alpha$, $\mathbf{h}_\beta$, $\mathbf{h}_\delta$ and vectors of values $\mathbf{x}_\alpha^{\mathbf{k}}$, $\mathbf{x}_\beta^{\mathbf{k}}$, $\mathbf{x}_\delta^{\mathbf{k}}$.
4) If $a > 1$, select two wolves $\rho 1$ and $\rho 2$, randomly among the herd of Q wolves, with $\rho 1 \neq \rho 2$. Store the corresponding vectors of indexes $\mathbf{h}_{\rho 1}$, $\mathbf{h}_{\rho 2}$ and vectors of values $\mathbf{x}_{\rho 1}^{\mathbf{k}}$, $\mathbf{x}_{\rho 2}^{\mathbf{k}}$.
   Else if $a \leq 1$, go to step 5.
5) Repeat steps for each wolf q, $q = 1, \dots, Q$, with vector of values $\mathbf{x}_q^{\mathbf{k}}(\text{iter})$ and vector of indexes $\mathbf{h}_q(\text{iter})$:
   a) Select leader $\mathbf{x}_{\text{l}}^{\mathbf{k}}$ (r real random number between 0 and 1):

if $a > 1$:

$$\mathbf{x}_{\text{l}}^{\mathbf{k}} = \begin{cases} \mathbf{x}_\alpha^{\mathbf{k}} & \text{if} \quad r \leq \frac{a}{10} \\ \mathbf{x}_\beta^{\mathbf{k}} & \text{if} \quad r > \frac{a}{10} \quad \text{and} \quad r \leq \frac{2a}{10} \\ \mathbf{x}_\delta^{\mathbf{k}} & \text{if} \quad r > \frac{2a}{10} \quad \text{and} \quad r \leq \frac{3a}{10} \\ \mathbf{x}_{\rho 1}^{\mathbf{k}} & \text{if} \quad r > \frac{3a}{10} \quad \text{and} \quad r \leq \frac{4a}{10} \\ \mathbf{x}_{\rho 2}^{\mathbf{k}} & \text{if} \quad r > \frac{4a}{10} \quad \text{and} \quad r \leq \frac{5a}{10} \\ \mathbf{x}_\alpha^{\mathbf{k}} & \text{if} \quad r > \frac{5a}{10} \end{cases}$$

if $a \leq 1$:

$$\mathbf{x}_{\text{l}}^{\mathbf{k}} = \begin{cases} \mathbf{x}_\alpha^{\mathbf{k}} & \text{if} \quad r \leq \frac{a}{6} \\ \mathbf{x}_\beta^{\mathbf{k}} & \text{if} \quad r > \frac{a}{6} \quad \text{and} \quad r \leq \frac{2a}{6} \\ \mathbf{x}_\delta^{\mathbf{k}} & \text{if} \quad r > \frac{2a}{6} \quad \text{and} \quad r \leq \frac{3a}{6} \\ \mathbf{x}_\alpha^{\mathbf{k}} & \text{if} \quad r > \frac{3a}{6} \end{cases}$$

   Store the vector of indexes corresponding to $\mathbf{x}_{\text{l}}^{\mathbf{k}}$ in a vector denoted by $\mathbf{h}_{\text{l}}$.
   b) Update the N indexes and the N values for wolf q ($i = 1, \dots, N$):

$$h_i(\text{iter}+1) = (h_i(\text{iter}) + \Delta \; sgn(h_i^{\text{l}} - h_i(\text{iter}))) \; mod \; H_i$$

   and

$$x_i(\text{iter}+1) = \mathbf{d}_i^{val}(h_i(\text{iter}+1))$$

   c) Get the updated vectors of indexes and values:
   $\mathbf{h}_q(\text{iter}+1) = [h_1(\text{iter}+1), \dots, h_i(\text{iter}+1), \dots, h_N(\text{iter}+1)]^T$.
   $\mathbf{x}_q^{\mathbf{k}}(\text{iter}+1) = [x_1(\text{iter}+1), \dots, x_i(\text{iter}+1), \dots, x_N(\text{iter}+1)]^T$.
6) Exchange the current population with the new one, obtained at step 5
7) If $\text{iter} < \text{T}_{\text{max}}$ or $f(\mathbf{x}_q^{\mathbf{k}}(\text{iter})) > \epsilon$, increase iter, and go to step 2.

Output: estimated parameter values $\hat{K_1}, \hat{K_2}, \dots, \hat{K_N}$

ISBN 978-90-827970-1-5 © EURASIP 2018

Function	f_{min}	$\mathbf{d}^{val}$	H_i
F_1	0	$[-100,\ldots,0,\ldots,100]^T$	201
F_2	0	$[-10,\ldots,0,\ldots,10]^T$	41
F_{16}	-1.0316	$[-5,\ldots,0,\ldots,5]^T$	101
F_{18}	3	$[-2,\ldots,0,\ldots,2]^T$	41

Table I: Benchmark functions: expected minimum, range and step for the search space

iteration $\mathrm{T}_{\max}$ for the best wolf, namely α.
The statistical mean is the average (Avg.) of fitness values:
$Avg = \frac{1}{M}\sum_{m=1}^{M} f(\mathbf{x}_\alpha^\mathbf{k})_m$
The standard deviation (Std.) is a representation for the variation of the obtained best solutions:
$Std = \sqrt{\frac{1}{M}\sum_{m=1}^{M}(f(\mathbf{x}_\alpha^\mathbf{k})_m - Avg)^2}$
The median (Med.) is the value separating the $\frac{M}{2}$ higher half from the $\frac{M}{2}$ lower half of values obtained for $f(\mathbf{x}_\alpha^\mathbf{k})$.
As other versions of GWO, our improved discrete GWO and MODGWO involve few parameters: we use $Q = 30$ search agents. With $\mathrm{T}_{\max} = 3000$ for the proposed improved discrete GWO and $\mathrm{T}_{\max} = 10000$ iterations for the MODGWO, the time required per run is always slightly lower with the proposed method than for MODGWO. Programs were written in $C++$, and executed on a PC running Windows, with a 3GHz double core and 3GB RAM. The results obtained for each algorithm are presented in Tables II and III.

F	Avg.	Std.	Med.	Time
F_1	3.233	2.046	2	289.83 ms
F_2	0.200	0.249	0	291.80 ms
F_{16}	-0.974	0.094	-1.0298	261.20 ms
F_{18}	4.628	2.576	3	237.17 ms

Table II: Results MODGWO with $T_{max} = 10000$

We notice that for slightly lower computational times, the proposed method outperforms MODGWO on the considered benchmark functions in terms of Avg., Std., and Med. Moreover, as indicated by the median value, our improved discrete GWO has found the expected minimum on all the benchmark functions on at least 50% of the runs.

F	Avg.	Std.	Med.	Time
F_1	0	0	0	279.67 ms
F_2	0	0	0	258.13 ms
F_{16}	-1.0298	$7.42\ 10^{-16}$	-1.0298	191.00 ms
F_{18}	3	$2.26\ 10^{-15}$	3	173.43 ms

Table III: Results improved discrete GWO with $T_{max} = 3000$

V. Conclusion

This paper proposes an improved discrete GWO, one of the first bio-inspired optimization algorithms which tackles discrete problems. The proposed method is tested on unimodal and multi-modal benchmark functions and yields results which outperform the existing MODGWO.

References

[1] P. Di Lorenzo, P. Banelli, and S. Barbarossa, "Optimal sampling strategies for adaptive learning of graph signals," in *Signal Processing Conference (EUSIPCO), 2017 25th European*. IEEE, 2017, pp. 1684–1688.

[2] P. Di Lorenzo, E. Isufi, P. Banelli, S. Barbarossa, and G. Leus, "Distributed recursive least squares strategies for adaptive reconstruction of graph signals," in *Signal Processing Conference (EUSIPCO), 2017 25th European*. IEEE, 2017, pp. 2289–2293.

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

[4] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46 – 61, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0965997813001853

[5] V. K. Kamboj, "A novel hybrid pso–gwo approach for unit commitment problem," *Neural Computing and Applications*, pp. 1–13, 2015.

[6] E. Emary, H. M. Zawbaa, and A. E. Hassanien, "Binary grey wolf optimization approaches for feature selection," *Neurocomputing*, vol. 172, pp. 371âĂŞ–381, 2016.

[7] C. Lu, S. Xiao, X. Li, and L. Gao, "An effective multi-objective discrete grey wolf optimizer for a real-world scheduling problem in welding production," *Advances in Engineering Software*, vol. 99, pp. 161 – 176, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0965997816301260

[8] C. Lu, L. Gao, X. Li, and S. Xiao, "A hybrid multi-objective grey wolf optimizer for dynamic scheduling in a real-world welding industry," *Engineering Applications of Artificial Intelligence*, vol. 57, pp. 61 – 79, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0952197616301968

[9] N. Mittal, U. Singh, and B. Singh Sohi, "Modified grey wolf optimizer for global engineering optimization," *Applied Computational Intelligence and Soft Computing*, vol. Article ID 7950348, p. 16 pages, 2016.

ISBN 978-90-827970-1-5 © EURASIP 2018