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 \( a \); matrices by boldface uppercase roman, like \( A \); 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 \( i \).
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 $\alpha$, $\beta$, $\delta$, and the $\omega$ wolves. Searching and attacking prey are implemented. The updated position is calculated as:

$$x^k_{q}(\text{iter} + 1) = \frac{1}{3} (y^\alpha + y^\beta + y^\delta)$$  \hspace{1cm} (1)

It results from the equal contribution of the $\alpha$, $\beta$, and the $\delta$ wolves. These contributions are computed as follows, for instance for the $\alpha$:

$$y^\alpha = x^k_{\alpha} - b \cdot d_{\alpha}$$  \hspace{1cm} (2)

with: $d_{\alpha} = |c \cdot x^k - x^k_{\text{iter}}|$

The vectors $b$ and $c$ are calculated as $b = 2a \cdot r_1 - a$ and $c = 2 - 2r_2$. In these expressions, the components of vector $a$ 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 $r_1$, $r_2$ 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 $a$. 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 $a$. In the seminal version of GWO [4], $a$ is regularly decreased from 2 to 0: $a = 2(1 - \frac{\text{iter}}{T_{\text{max}}})$, where $\text{iter}$ is the iteration index, and $T_{\text{max}}$ 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 $a$:

$$a = 2(1 - \frac{\text{iter}^2}{T_{\text{max}}^2})$$  \hspace{1cm} (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 $r$ be a random real number between 0 and 1.

$$K_1 = \begin{cases} K_0^\alpha & \text{if } r \leq \frac{1}{3} \\ K_1^\beta & \text{if } \frac{1}{3} < r \leq \frac{2}{3} \\ K_2^\delta & \text{if } r > \frac{2}{3} \end{cases}$$  \hspace{1cm} (4)

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

$$K_2 = \begin{cases} K_2 \cdot K_2^\beta \cdot K_2^\delta & \text{if } r \leq a \\ K_2^\beta & \text{if } r > a \end{cases}$$  \hspace{1cm} (5)

In Eq. (5), $a = 1 - \frac{\text{iter}}{T_{\text{max}}}$, and $K_2^\beta$ 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 $\text{iter}$ is denoted by $x^k_{q}(\text{iter})$ 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 $d_{\text{val}} = [K_1^1, \ldots, K_i^{h_i}, \ldots, K_N^{H_i}]^T$ is the 'search domain' for $K_i$, that is, it contains all
acceptable values for parameter $K_i$.

- Vector $d_i^{ind} = [1, \ldots, h_i, \ldots, H_i]^T$ contains the index of each acceptable value for the $i$th parameter.
- $d_i^{ind}$ contains $H_i$ integer values in increasing order.
- A wolf denoted by $x^k_q(\text{iter})$, is a vector with $N$ components that we call vector of ‘values’. These values are denoted as follows: $x^k_q(\text{iter}) = [x_1(\text{iter}), \ldots, x_i(\text{iter}), \ldots, x_N(\text{iter})]^T$, where the index $q$ is omitted for sake of clarity.
- Vector $h_q(\text{iter})$ is the vector of $N$ indexes associated with wolf $x^k_q(\text{iter})$. These values are denoted as follows: $h_q(\text{iter}) = [h_1(\text{iter}), \ldots, h_i(\text{iter}), \ldots, h_N(\text{iter})]^T$, where the index $q$ is omitted for sake of clarity.
- Vectors $x^k_1$, $x^k_2$, $x^k_3$, $x^k_4$, $x^k_5$, $x^k_6$ are the vectors of values for leaders $\alpha$, $\beta$, $\delta$, and for two wolves $\rho_1$ and $\rho_2$, selected at random among the $Q$ wolves of the population.
- Vectors $h_1$, $h_2$, $h_3$, $h_4$, $h_5$, $h_6$ are the vectors of indexes for leaders $\alpha$, $\beta$, $\delta$, and for the two random wolves $\rho_1$ and $\rho_2$.
- The current leader in our iterative algorithm is denoted by $l$. Its vector of values is $x^k = [x^k_1, \ldots, x^k_i, \ldots, x^k_N]^T$ and its vector of indexes is $h^k = [h^k_1, \ldots, h^k_i, \ldots, h^k_N]^T$.
- The scalar $x^k_i$ is the $i$th component of $x^k$, and the scalar $h^k_i$ denotes the $i$th component of $h^k$. We notice that a value located at the component with index $h_i$ in $d_i^{ind}$ is denoted by $K^{h_i}_i$. For instance, for the vector of values $x^k_q(\text{iter}) = [K^1_i, \ldots, K^N_i]^T$, the associated vector of indexes is $h_q(\text{iter}) = [2, \ldots, 5]^T$.

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 $x^k$ and the update process of the index vector $h_q(\text{iter})$.

As the parameter $a$ is decreasing from 2 to 0 across the iterations, it is more and more probable for $a$ 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, $\text{sgn}(\cdot)$ denotes the sign function, and $\text{mod}$ denotes the ‘Modulo’ operator. $\Delta$ is computed as follows:

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

where $\phi$ 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 exploitation is emphasized when $a > 1$ at the beginning of the process, and exploration is emphasized when $a \leq 1$ at the end of the process. The proposed algorithm has a complexity of $O(T_{\text{max}}.N.Q)$.

### 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]:

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

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

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

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

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, $f(x^{k}_m)$, the fitness value obtained at
Algorithm 1 Pseudo-code: Improved Discrete Grey Wolf Optimization for multiple parameter estimation

**Inputs:** fitness function \( f \), number \( N \) of expected parameters, small factor \( \epsilon \) set by the user, to stop the algorithm, maximum number of iterations \( T_{\text{max}} \).

For each parameter indexed by \( i = 1, \ldots, N \): the search space \( \mathbf{d}_{i}^{\text{ind}} \) with \( H_i \) possible values.

1) Set iteration number \( \text{iter} = 1 \), create an initial set of index vectors \( \mathbf{h}_q(\text{iter}), q = 1, \ldots, Q \). For each index \( q \), each component \( h_i^{\text{iter}} \) is an integer value between 1 and \( H_i \).

Create an initial herd composed of \( Q \) wolves \( \mathbf{x}_q^k(\text{iter}), q = 1, \ldots, Q \) with the \( N \) required parameter values. This initial population takes the form of a matrix with \( Q \) rows and \( N \) columns. For each index \( q, x_i(\text{iter}) = \mathbf{d}_{i}^{\text{ind}}(h_i(\text{iter})) \).

2) Evaluate fitness function value \( f(\mathbf{x}_q^k(\text{iter})) \) of each wolf \( \mathbf{x}_q^k(\text{iter}), q = 1, \ldots, Q \).

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 \( \mathbf{x}_\alpha^k, \mathbf{x}_\beta^k, \mathbf{x}_\delta^k \).

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 \( \mathbf{x}_{\rho 1}^k, \mathbf{x}_{\rho 2}^k \).

Else if \( a \leq 1 \), go to step 5.

5) Repeat steps for each wolf \( q \), \( q = 1, \ldots, Q \), with vector of values \( \mathbf{x}_q^k(\text{iter}) \) and vector of indexes \( \mathbf{h}_q(\text{iter}) \):

   a) Select leader \( \mathbf{x}_1^k \) (\( r \) real random number between 0 and 1):

      if \( a > 1 \):

      \[
      \mathbf{x}_1^k = \begin{cases} 
        \mathbf{x}_0^k & \text{if } r \leq \frac{a}{10} \\
        \mathbf{x}_1^k & \text{if } r > \frac{a}{10} \text{ and } r \leq \frac{2a}{10} \\
        \mathbf{x}_2^k & \text{if } r > \frac{2a}{10} \text{ and } r \leq \frac{3a}{10} \\
        \mathbf{x}_3^k & \text{if } r > \frac{3a}{10} \text{ and } r \leq \frac{4a}{10} \\
        \mathbf{x}_a^k & \text{if } r > \frac{4a}{10} \text{ and } r \leq \frac{5a}{10} \\
      \end{cases}
      \]

      Store the vector of indexes corresponding to \( \mathbf{x}_1^k \) in a vector denoted by \( \mathbf{h}_1 \).

   b) Update the \( N \) indexes and the \( N \) values for wolf \( q (i = 1, \ldots, N) \):

      \[
      h_i(\text{iter} + 1) = (h_i(\text{iter}) + \Delta \sgn(h_i^1 - h_i(\text{iter}))) \mod H_i
      \]
      \[
      x_i(\text{iter} + 1) = d_i^{\text{val}}(h_i(\text{iter} + 1))
      \]

   c) Get the updated vectors of indexes and values:

      \[
      \mathbf{h}_q(\text{iter} + 1) = [h_1(\text{iter} + 1), \ldots, h_i(\text{iter} + 1), \ldots, h_N(\text{iter} + 1)]^T
      \]
      \[
      \mathbf{x}_q^k(\text{iter} + 1) = [x_1(\text{iter} + 1), \ldots, x_i(\text{iter} + 1), \ldots, x_N(\text{iter} + 1)]^T
      \]

6) Exchange the current population with the new one, obtained at step 5.

7) If \( \text{iter} < T_{\text{max}} \) or \( f(\mathbf{x}_q^k(\text{iter})) > \epsilon \), increase \( \text{iter} \), and go to step 2.

**Output:** estimated parameter values \( \hat{K}_1, \hat{K}_2, \ldots, \hat{K}_N \)
iteration $T_{\text{max}}$ for the best wolf, namely $\alpha$.

The statistical mean is the average (Avg.) of fitness values:

$$\text{Avg} = \frac{1}{M} \sum_{m=1}^{M} f(x_{\text{opt}, m})$$

The standard deviation (Std.) is a representation for the variation of the obtained best solutions:

$$\text{Std} = \sqrt{\frac{1}{M} \sum_{m=1}^{M} (f(x_{\text{opt}, m}) - \text{Avg})^2}$$

The median (Med.) is the value separating the $\frac{M}{2}$ higher half from the $\frac{M}{2}$ lower half of values obtained for $f(x_{\text{opt}, m})$.

As other versions of GWO, our improved discrete GWO and MODGWO involve few parameters: we use $Q = 30$ search agents. With $T_{\text{max}} = 3000$ for the proposed improved discrete GWO and $T_{\text{max}} = 10000$ 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 dual core and 3GB RAM. The results obtained for each algorithm are presented in Tables II and III.

<table>
<thead>
<tr>
<th>Function</th>
<th>$f_{\text{min}}$</th>
<th>$d^{\text{val}}$</th>
<th>$H_{i}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_1$</td>
<td>0</td>
<td>[-100, . . . , 0, . . . , 100]</td>
<td>201</td>
</tr>
<tr>
<td>$F_2$</td>
<td>0</td>
<td>[-10, . . . , 0, . . . , 10]</td>
<td>41</td>
</tr>
<tr>
<td>$F_{16}$</td>
<td>-1.0316</td>
<td>[-5, . . . , 0, . . . , 5]</td>
<td>101</td>
</tr>
<tr>
<td>$F_{18}$</td>
<td>3</td>
<td>[-2, . . . , 0, . . . , 2]</td>
<td>41</td>
</tr>
</tbody>
</table>

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

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.

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

<table>
<thead>
<tr>
<th>$F$</th>
<th>Avg.</th>
<th>Std.</th>
<th>Med.</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_1$</td>
<td>0.233</td>
<td>2.046</td>
<td>2</td>
<td>289.83 ms</td>
</tr>
<tr>
<td>$F_2$</td>
<td>0.200</td>
<td>0.249</td>
<td>0</td>
<td>291.80 ms</td>
</tr>
<tr>
<td>$F_{16}$</td>
<td>-0.974</td>
<td>0.094</td>
<td>-1.0298</td>
<td>261.20 ms</td>
</tr>
<tr>
<td>$F_{18}$</td>
<td>4.028</td>
<td>2.576</td>
<td>3</td>
<td>237.17 ms</td>
</tr>
</tbody>
</table>

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

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