Conference Paper Title*

Han Li
School of Information Science and Engineering
Yunnan University
Kunming,China
lihan@mail.ynu.edu.cn

Weidong Li
School of Mathematics and Statistics
Yunnan University
Kunming,China
weidongmath@126.com

Xuejie Zhang*
School of Information Science and Engineering
Yunnan University
Kunming,China
xjzhang@ynu.edu.cn

Abstract—Natural phenomena are the source of all heuristic algorithms. For example, particle swarm optimization(PSO) algorithm comes from the foraging behavior of population [1]. The dwarf mongoose optimization algorithm (DMO) simulates the foraging behavior of the dwarf mongoose. Simulated annealing algorithms (SA) is very similar to the physical system annealing process [3]. However, many heuristic algorithms can only solve continuous problems, because them more or less imitated gradient. For discrete problems, there is no good solution, which severely limits the use of these algorithms in practical engineering problems. Because discrete decision variables have practical physical significance. In this paper, we use genetic algorithms (GA) and simulated annealing algorithms for reference, and improve the dwarf mongoose optimization algorithm according to the physical background of the problem to enable it to solve physical problems

Index Terms—component, formatting, style, styling, insert

I. Introduction

This document is a model and instructions for LaTeX. Please observe the conference page limits.

II. An introduction to Dwarf Mongoose Optimization Algorithm

The DMO algorithm is inspired by unique compensatory behavioral adaptations of the dwarf mongoose [2]. Dwarf mongoose is a carnivore in African, although this animal is small in size. They perform foraging and other behaviors through division of labor, so they engage in many compensatory behaviors.

Unlike humans, which are a patriarchal society, dwarf mongoose are a matriarchal society. It is not the female that takes care of the pups, but usually a certain number of individuals are selected to act as babysitters.

The algorithm divide the population into three groups: the alpha group, babysitters, and the scout group. When the alpha group go out to look for food, the young and babysitters stay at the nest. Note that looking for food means exploring space.

A. Babysitters

The babysitters don't go out foraging, they stay with the young. To be fair, babysitters are rotated periodically, and other individuals in the population assume the role of babysitters. and the number of them depends on the the number of individuals. In our experiments, we set the number of nannies to 3%.

B. Alpha group

The number of individuals in the population is n, X^t denotes all individuals in the population at time t, X_i^t represents the ith individual in the population at that time. In every iteration, the probability value for each population which is select as a alpha female is calculated is as follows:

$$\alpha = \frac{f_i^t}{\sum_{i=1}^{n} f_i^t} \tag{1}$$

And the alpha group is updated in the following way:

$$X^{t+1} = X^t + phi * \text{ peep} \tag{2}$$

where, phi follows a uniform distribution $phi \sim U[-1, 1]$. In every iteration, the sleeping mound is calculated as the Equation 3, and the average value of the sleeping mound is calculated as the Equation 4.

$$sm_i^t = \frac{f_i^{t+1} - f_i^t}{\max\{|f^{t+1}, f_i^t|\}} \tag{3}$$

$$\varphi = \frac{\sum_{i=1}^{n} sm_i}{n} \tag{4}$$

C. Scout group

The scouts find the next sleeping mound, and the mongooses do not come back to previous sleeping mound, which can ensure no overgrazing. In other words, this behavior ensures exploration validity. Equation 5 imitated this behavior.

$$X_i^{t+1} = \begin{cases} X_i^t - CF * \text{ phi } * \text{ rand } * \left[X_i^t - \overrightarrow{\boldsymbol{M}}\right], \\ \qquad\qquad \text{if } \varphi_{i+1} > \varphi_i \\ X_i^t + CF * \text{ phi } * \text{ rand* } \left[X_i^t - \overrightarrow{\boldsymbol{M}}\right], \\ \qquad\qquad \text{else} \end{cases} \tag{5}$$

where, $rand$ is a random number between 0 and 1, CF is the parameter which can controls the collective-volitive movement of the mongoose and is calculated in Equation 6. And $\overrightarrow{\boldsymbol{M}} = \sum_{i=1}^{n} \frac{X_i \times sm_i}{X_i}$.

$$CF = \left(1 - \frac{iter}{Max_{iter}}\right)^{\left(2 \frac{i \text{ iter}}{Max_{iter}}\right)} \tag{6}$$

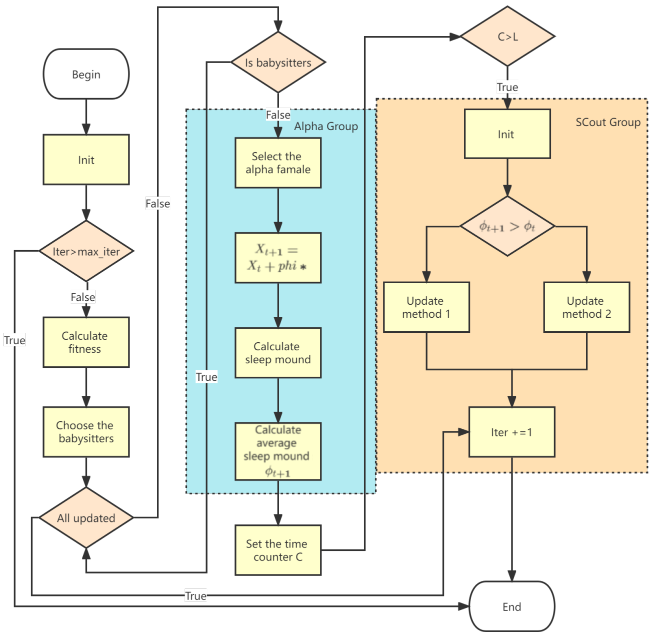

Fig. 1: Flow chart of genetic algorithm

D. Description of DMO algorithm

The flow chart of this algorithm is shown in Figure 1. When a population is initialized, the algorithm begins. When the number of iterations is less than the maximum number of iterations, calculating the fitness. And next step is to choose babysitters, who are not involved in foraging. Then perform update of alpha group. All but babysitters in the population were updated. At the time of renewal, a female was first selected using Equation 1, and using Equation 2 to update this individual. We need to calculate sleep mound and the average of the sleep mound in order for scouts group to use. The final step of alpha group is setting the time counter$C$.

III. Discrete Dwarf Mongoose Optimization Algorithm

The key step in designing a discrete algorithm is to define the representation of the solution. Because each decision variable has a real physical meaning.

In designing the update operation, we use the update method of genetic algorithm for reference. The GA which is based on Darwin's theory of natural selection which is published by John Holland in 1976 [4] is a smart heuristic algorithm that can be used to solve integer programming. However, the genetic algorithm does not consider the actual physical meaning of the decision variables, but only encodes variables to solve discrete problems. it draws on the behavior of chromosomes in cells. In Darwin's theory of natural selection, individuals adapted to the environment survive and reproduce. In the process of biological reproduction, chromosome experience crossover and mutation. Because GA need to simulate the behavior of chromosomes, we need to encode and decode every iteration which requires a lot of calculation, as shown in Figure 2.

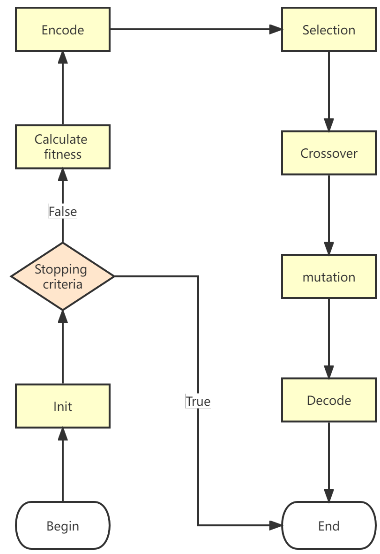

Fig. 2: Flow chart of genetic algorithm

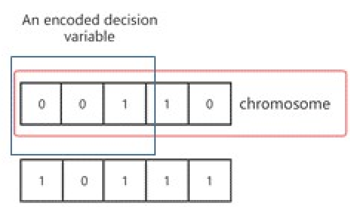

Fig. 3: Coding operations in GA

When encoding the variable, the number of bits encoded is determined by the upper bound, the lower bound, and the interval. The formula is as follows:

$$m = \lceil \log_2 \frac{ub - lb}{p} + 1 \rceil \tag{7}$$

Where ub is the upper bound, lb is the lower bound, and p is the interval.

As shown in the Figure 3, in genetic algorithm, the first three binary digits form a decision variable. When updating a variable , only the encoded string needs to be operated like a logical operation.

Ternary Operators$(a * X_1 - X_2)$

Our problem is homogeneous with the parallel machines scheduling problem. We will describe the problem in two ways, in order to better understand the representation of the decision variables.

A. The Parallel Machines Scheduling Problem

The description of it is as follows: there are n jobs $J = \{1, \ldots, n\}$, m machines which are identical $M = \{1, \ldots, m\}$. Each job j is independent. Every job should be finished on one of that machines, and the time which the job i need on the machine for processing job is p_i. Once the jobs are placed on the machine, they need to work until it is completed. As a result, they can not be interrupted. Such that the maximum running time of all machines is minimized.

The parallel machines scheduling problem can be formulated as an integer programming problem:

$$\min \max_{1 \le j \le m} \sum_{i=1}^{n} p_i x_{ij} \tag{8}$$

$$\text{s.t.} \sum_{j=1}^{m} x_{ij} = 1 \quad i = 1, \ldots, n \tag{9}$$

$$x_{ij} \in \{0, 1\} \tag{10}$$

Where the Equation (8) shows that we should minimize the maximum completion time. The Equation (9) shows each job must be assigned to a machine. If $x_{ij} = 1$, it means that job i is assigned to j's machine.

However, the spatial complexity of this formulation is $O(mn)$, and because each variable is a binary variable, so the time complexity is $O(2^{mn})$. We formulate the parallel machines scheduling as another form.

$$\min \max_{1 \le j \le m} \sum_{1 \le i \le n, x_i = j} p_i \tag{11}$$

$$\text{s.t.} \quad x_i \in \{0, 1, \ldots, m\}, \quad i = 1, \ldots, n \tag{12}$$

Where the spatial complexity of this formulation is $O(n)$, and the time complexity is $O(m^n)$

B. Different From GA

C. Our Algorithm

In our algorithm, we learned from Mendel's genetic law. We define the decision variables of individuals with higher fitness as dominant traits. In Mendel's experiment, the first generation of tall and short stemmed peas produced offspring with tall stems. Similarly, we define that dominant traits cover hidden traits.

We define a ternary operator ($phi * [X_a + X_b]$), $phi \in (0, 1)$ is a number that controls the variation, and X_a, X_b are two vectors of the same dimension. At first, we compute $X_c = phi * X_a + (1 - phi) * X_b$, for each element in X, the probability of being selected is calculated by the following formula:

$$X_c = phi * X_a + (1 - phi) * X_b \tag{13}$$

$$p(X_{ai}) = \frac{|X_{bi} - X_{ci}|}{|X_{bi} - X_{ai}|} \tag{14}$$

Algorithm 1: Discrete Dwarf Mongoose Optimization Algorithm

```
Input: Parameters n
Output: Best solution
Initialize the algorithm parameters:[peep]
Initialize the mongoose populations (search agents): n
Initialize the number of babysitters: bs
Set babysitter exchange parameter L
for iter = 1 : m do
    for k = 1 : bs do
        Select the babysitters
    end
    Calculate the fitness of the mongoose
    Set time counter C
    Find the alpha based on the equation:
        α = fit_i / Σ_{i=1}^{n} fit_i
    Produce a candidate food position based on the equation:
        X_i^{t+1} = phi * [X_i^t + X_peep^t]
    Evaluate new fitness of X_{i+1}
    Evaluate sleeping mound
        sm_i = |fit_{i+1} − fit_i| / max{|fit_{i+1}, fit_i|}
    Compute the average value of the sleeping mound found:
        φ = Σ_{i=1}^{n} sm_i / n
    Compute the movement vector using
        M⃗ = X_cbest
    Exchange babysitters if C ≥ L, and set initialize bs position and calculate fitness fit_i ≤ α
    Simulate the scout mongoose next position using
        X_{i+1} = (CF * phi * rand) * [X_i + M⃗]   if φ_{i+1} > φ_i
                  (CF * phi * rand) * [M⃗ + X_i]   else
end
return best solution
```

Where, in line 11: $\alpha = \frac{fit_i}{\sum_{i=1}^{n} fit_i}$; line 12: $X_i^{t+1} = phi * [X_i^t + X_{peep}^t]$; line 14: $sm_i = \frac{|fit_{i+1} - fit_i|}{\max\{|fit_{i+1}, fit_i|\}}$; line 15: $\varphi = \frac{\sum_{i=1}^{n} sm_i}{n}$; line 16: $\overrightarrow{M} = X_{cbest}$; line 18:

$$X_{i+1} = \begin{cases} (CF * phi * rand) * [X_i + \overrightarrow{M}] & \text{if } \varphi_{i+1} > \varphi_i \\ (CF * phi * rand) * [\overrightarrow{M} + X_i] & \text{else} \end{cases}$$

IV.

V. Prepare Your Paper Before Styling

Before you begin to format your paper, first write and save the content as a separate text file. Complete all content and organizational editing before formatting.

Please note sections ??–?? below for more information on proofreading, spelling and grammar.

Keep your text and graphic files separate until after the text has been formatted and styled. Do not number text heads—LaTeX will do that for you.

Acknowledgment

The preferred spelling of the word "acknowledgment" in America is without an "e" after the "g". Avoid the stilted expression "one of us (R. B. G.) thanks ...". Instead, try "R. B. G. thanks...". Put sponsor acknowledgments in the unnumbered footnote on the first page.

References

[1] James Kennedy and Russell Eberhart. "Particle swarm optimization". In: Proceedings of ICNN'95-international conference on neural networks. Vol. 4. IEEE. 1995, pp. 1942–1948.

[2] Jeffrey O Agushaka, Absalom E Ezugwu, and Laith Abualigah. "Dwarf mongoose optimization algorithm". In: Computer methods in applied mechanics and engineering 391 (2022), p. 114570.

[3] Peter JM Van Laarhoven and Emile HL Aarts. "Simulated annealing". In: Simulated annealing: Theory and applications. Springer, 1987, pp. 7–15.

[4] Jeffrey R. Sampson. "Adaptation in Natural and Artificial Systems (John H. Holland)". In: SIAM Review 18.3 (1976), pp. 529–530. doi: 10.1137/1018105. eprint: https://doi.org/10.1137/1018105. url: https://doi.org/10.1137/1018105.