Grey Wolf Optimizer

Jan Vykruta

February 2021

Introduction

In this article we would like to outline a PSO algorithm based on the hierarchy and hunting of grey wolves. The algorithm is described in greater depth in [1]. In Section 1 we describe the social hierarchy of grey wolves. In Section 2 we formulate the algorithm and mathematical model. In Section 3 we mention our results. Finally, in Section 4 we provide link to our implementation as well as some comments regarding the code.

1 Grey wolves

Grey wolf packs are formed of alphas, betas, deltas and omegas. Alphas are the leaders of a pack, they are the the most adept at leading a pack though they are not necessarily the strongest. There is one male and one female, and they are the only ones that mate. Betas are second only to alphas. They usually take place of alphas if something were to happen to them and the pack would need new leaders. Omegas are the lowest in the pack hierarchy. They are useful to vent violence and frustration of other wolves. Finally, all other wolves are deltas. They have to submit to alphas and betas, however, they are above omegas. Scouts, hunters, elders and caretakers belong to this group. Hunting consists of the following phases:

- Tracking, chasing and approaching the prey.
- Pursuing, encircling and harassing the prey.
- Attack towards the prey.

2 Algorithm

In the algorithm each wolf represents a solution to an optimization problem. Alpha, beta and delta represent the first three best solutions, other solutions are omegas that follow the leaders.

The wolves encircle the prey. The following equations mathematically model the situation:

$$D = |C * X_p(t) - X(t)| \tag{1}$$

$$X(t+1) = X_p(t) - A * D \tag{2}$$

$$A = 2a * r_1 - a \tag{3}$$

$$C = 2r_2 \tag{4}$$

where t indicates the current iteration, X_p is the position of the prey, X is the position of a wolf, a is linearly decreased from 2 to 0 over the course of iterations, and r_1 and r_2 are random vectors in $[0, 1]$.

Using Equations 1 and 2 wolves update their position randomly while moving in a hyper-cube around the prey.

While hunting wolves can recognize the location of the prey and encircle it, in an optimization problem we do not know the location of the prey (optimum). Therefore, to simulate the behaviour we suppose that alpha, beta and delta have a good knowledge of the location of the prey, and we oblige the other wolves (omegas) to update their position according to the leaders. The situation is modeled as follows:

$$D_i = |C_i * X_i - X|, i \in \{\alpha, \beta, \delta\} \tag{5}$$

$$W_i = X_i - A_i * D_i, i \in \{\alpha, \beta, \delta\} \tag{6}$$

$$X(t+1) = \frac{W_\alpha + W_\beta + W_\delta}{3} \tag{7}$$

where Equations 5 and 6 are an application of the Equations 1 and 2 where the position of the prey is replaced by a position of a leader, X_i are positions of the leaders, C_i and A_i are distinct instances of Equations 3 and 4 (for each instance new r_1 and r_2 are generated).

The final position of each wolf will be a random position within a hyper-sphere which is defined by

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
$f_7(x) = \sum_{i=1}^{n} i * x_i^4 + random[0, 1)$	30	$[-1.28, 1.28]$	0
$f_{11}(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$	30	$[-600, 600]$	0

Table 1: Functions used in our benchmark. Numbering corresponds to the original paper.

the positions of the leaders (the leaders estimate the position of the prey or optimum). Please note, that all wolves including the leaders update their position in the Equation 7.

Exploitation is achieved by A which is in the interval $[-4, 4]$ and the interval tightens as the algorithm progresses. When $|A| < 1$ the wolves attack towards the prey.

Exploration is achieved by both A and C. When $|A| > 1$ the wolves diverge from the prey in hope of finding a better candidate. This occurs mostly in the beginning of the algorithm, however, that could result in being trapped in a local optima in later stages of the algorithm. Therefore, the interval of C remains the same throughout a run of the algorithm and serves to avoid being trapped in a local optima. C can be considered as the effect of obstacles in nature which prevent the wolves in directly approaching the prey.

3 Results

We have tested the implementation on four benchmark functions which can be seen in Table 2. The function numbering corresponds to that in the original paper. We conclude that the implementation is correct as it provides results comparable to the original paper. In the original paper the algorithm is compared to several other PSO algorithms. The GWO algorithm provides competitive results both in classical benchmarks as well as in real engineering problems.

4 Implementation

Our implementation is available in a `git` repository[1]. Of particular interest can be `gwo.py` which implements the algorithm. New functions can be added to `functions.py` and appended to the `FUNCTIONS` list. The implementation requires `Python3` and packages `numpy` and `gnuplotlib`, additionally `gnuplotlib` requires `gnuplot`. If it is not available it is possible to run the optimizer without plotting by using the `-no-gnuplot` option. Instructions to run the optimizer are in the `README.md` file.

[1] https://github.com/JanVykruta/wolves_pso

Conclusion

We have implemented the GWO algorithm as described in the original paper and verified that it provides results comparable to those reported in the paper on a small subset of functions used by the authors.

References

[1] Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. Grey wolf optimizer. *Advances in Engineering Software*, 69:46 – 61, 2014.