Shark Smell Optimization

Jochem Ram (S2040328), Jerry Schonenberg (S2041022), Wouter van Tol (S2041340), Thomas Wink (S1705148)

Leiden Institute of Advanced Computer Science, The Netherlands

Abstract. In this paper a pseudocode is presented for the Shark Smell Optimization algorithm.

1 Introduction

For as long as humans exist, we have looked at nature for inspiration. Most inventions humans have made come from investigating natural phenomena and trying to reproduce what we see. This is still the case nowadays, where we think we know almost everything. It has proven useful to examine the way processes happen in nature and to try to implement this is problems we have ourselves. One example is to look at the way a shark, one of the best hunters in the sea, finds a pray in an efficient way. This can be used in an optimization problem to quickly find the best solution to a complex problem in a high dimensional solution space.
In this paper we present the findings of this shark-smell-optimization(SSO) algorithm. This will be done using IOHprofiler by comparing the performance of the algorithm for different objective functions.
The algorithm was originally introduced here [1].

2 Algorithm Description

This algorithm uses a "shark", an individual, to find a "wounded fish", the maximal value of the objective function, by following the "blood particles", the gradient of the objective function, and performing "circular searches", local searches around the individual's position at each stage. This is repeated for each individual until stage k_{max} is reached. The number of individuals and stages can be decided by the user.
A single iteration of the algorithm will first contain the calculation of the movement of the individuals in each dimension of the search space. Then a new position is calculated for every position. From this position a random local search will be done, where the best point from this search will be compared with the position from the the first search. The best one of these will be the new starting position for this individual for the next iteration.

2.1 Comparison with standard algorithms

The SSO algorithm is similar to simulated annealing. This algorithm is based on the concept of heat treatment in metals, in which a hot metal is slowly heated to let the atoms slowly settle to minimal energy states. This is used in optimization by perturbing each point randomly and choosing whether to move to that point or not. If the objective function in that point is higher, if will always move. If the objective function is lower, the chance is calculated with the difference in

the objective function values.
This algorithm matches the closest to the SSO because, unlike almost all natural optimization algorithms, these two algorithms are not based on mutation or swarm behavior. They are based on the searches of individual solutions without any communication between them. Also, both algorithms increase their stochastic search value. SSO does this by choosing random values around the solution candidate and evaluating all of them at once and simulated annealing does this by choosing a random value around the solution candidate and evaluating immediately. For simulated annealing this leads to a higher chance of reaching a local minimum, but this is mitigated by offering the chance of a restart where an individual is placed back to a previously found significantly better solution candidate.
However, where the SSO uses the gradient to find the movement direction of the search point, simulated annealing uses difference in objective function values in two points. Also, SSO can not move to a position with a lower objective function value while for simulated annealing this is one of it's strengths.

3 Pseudo-code

We followed the following notation convention. All user assigned variables are mentioned in Appendix A.

3.1 Variable explanation

- N: The number of individuals in the set/array.
- M: The dimensionality of the search space.
- k_{max}: Maximal amount of stages.
- O: Number of rotational candidate solutions.
- n, m, k and o: Administrative array counters for N, M, k_{max} and O respectively.
- X_n^k: The solution candidate $\mathbb{R}^M$ of an individual at a stage.
- V_n^k: The calculated movement in $\mathbb{R}^M$ for an individual at a stage.
- $v_{n,m}^k$: The calculated movement in a direction for an individual at a stage.
- Y_n^k: The provisional solution candidate in $\mathbb{R}^M$ after movement of an individual at a stage.
- Z_n^k: Set of O provisional solution candidates in $\mathbb{R}^M$ of an individual at a stage after local search.
- x_m: A variable in the search space.
- $x_m^{\min}$, $x_m^{\max}$: The minimal and maximal values in the search space for a variable.
- $f(\mathbf{x})$: Objective function value of $\mathbf{x}$ ($f : \mathbb{R}^M \to \mathbb{R}$).
- $\leftarrow$: Assignment operator.

- $\boldsymbol{\alpha}$: User assigned vector in $\mathbb{R}^k_{max}$ between 0 and 1. Represents the inertia coefficient for each stage.
- $\boldsymbol{\beta}$: User assigned vector in $\mathbb{R}^k_{max}$ between 0 and 1. Represents the velocity limiter ratio for each stage.
- $\boldsymbol{\eta}$: User assigned vector in $\mathbb{R}^k_{max}$ between 0 and 1. Limits the gradient of the objective function for each stage.
- $r1$ and $r2$: Random uniform value between 0 and 1 to give the algorithm more stochastic search value.
- $R3$: Vector in $\mathbb{R}^M$ with random uniform values between -1 and 1 to give the algorithm more stochastic search value.
- $\arg\max()$: Takes the variable with the highest objective function value.
- $\arg\max_{a=1}^{A}()$: Iterates over a and takes the variable with the highest objective function value.
- Further notation convention is based on mathematics.

3.2 Used functions

1) $v^k_{n,m} \leftarrow \eta_k \cdot R1 \cdot \frac{\partial f(x)}{\partial x_m}\Big|_{X^k_n} + \alpha_k \cdot R2 \cdot v^{k-1}_{n,m}$

This function is the calculation of movement in dimension m for individual n in stage k. In the first term of the calculation we take the partial derivative of the objective function to x_m, and then multiply it with a velocity limiter constant and a random value for extra stochastic search value. In the second term we take the movement from the last stage in this dimension for this individual and multiply it with a random value and an inertia constant. This function is used in line 16.

2) $\left|v^k_{n,m}\right| > \left|\beta_k \cdot v^{k-1}_{n,m}\right|$

This line limits the acceleration of the movement of an individual in dimension m. This is done by comparing the new velocity to the velocity in the previous stage multiplied with a velocity limiter constant. This function is used in line 17.

3) $Y^{k+1}_n \leftarrow X^k_n + V^k_n$

Assignment of provisional new location based on previous location and movement. This function is used in line 23.

4) $Z^{k+1,o}_n \leftarrow Y^{k+1}_n + R3 \cdot V^k_n$

This function finds O new solution candidates in a local search in the vicinity of Y^{k+1}_n. This is done with the provisional new position Y, and the velocity of this stage multiplied with a random vector R3. This function is used in line 27.

5) $X_n^{k+1} \leftarrow \arg\max(\arg\max_{o=1}^{O}(f(Z_n^{k+1,o})), f(Y_n^{k+1}))$

This function finds the best new location by comparing the provisional new location with the solution candidates from the local search, and testing their objective function value. This function is used in line 31.

3.3 Pseudocode

Algorithm 1 Shark Smell Optimization

```
1:  $M \leftarrow User \quad assigned$                                    ▷ Initialize
2:  $k_{max} \leftarrow User \quad assigned$
3:  $k \leftarrow 1$
4:  $O \leftarrow User \quad assigned$
5:  $\boldsymbol{\alpha} \leftarrow User \quad assigned$
6:  $\boldsymbol{\beta} \leftarrow User \quad assigned$
7:  $\boldsymbol{\eta} \leftarrow User \quad assigned$
8:  for $n = 1 \rightarrow n$ do
9:      for $m = 1 \rightarrow m$ do
10:         $x^1_{n,m} \leftarrow \boldsymbol{\mathcal{U}}(x^{\min}_m, x^{\max}_m)$
11:     end for
12: end for
13: while $k \leq k_{max}$ do
14:     for $n = 1 \rightarrow N$ do                 ▷ Calculation of movement per individual
15:         for $m = 1 \rightarrow M$ do
16:             $v^k_{n,m} \leftarrow \eta_k \cdot R1 \cdot \frac{\partial f(x)}{\partial x_m}\Big|_{X^k_n} + \alpha_k \cdot R2 \cdot v^{k-1}_{n,m}$
17:             if $\left|v^k_{n,m}\right| > \left|\beta_k \cdot v^{k-1}_{n,m}\right|$ then
18:                 $\left|v^k_{n,m}\right| \leftarrow \left|\beta_k \cdot v^{k-1}_{n,m}\right|$
19:             end if
20:         end for
21:     end for
22:     for $n = 1 \rightarrow N$ do        ▷ Provisional assignment of new position per individual
23:         $Y^{k+1}_n \leftarrow X^k_n + V^k_n$
24:     end for
25:     for $n = 1 \rightarrow N$ do    ▷ Random local search from provisional position per individual
26:         for $o = 1 \rightarrow O$ do
27:             $Z^{k+1,o}_n \leftarrow Y^{k+1}_n + R3 \cdot V^k_n$                     ▷ Deviation
28:         end for
29:     end for
30:     for $n = 1 \rightarrow N$ do        ▷ Definitive assignment of new position per individual
31:         $X^{k+1}_n \leftarrow \max(\max^O_{o=1}(f(Z^{k+1,o}_n)), f(Y^{k+1}_n))$
32:     end for
33:     $k \leftarrow k + 1$                                         ▷ Administration
34: end while
```

3.4 Assumptions

With this algorithm we have done the assumed that the objective function should be maximized. In our pseudocode, on line 27, we deviated from the original paper. In the original paper this line was:
$Z_i^{k+1,m} \leftarrow Y_i^{k+1} + R3 \cdot Y_i^{k+1}$
We changed the second Y to a V, as we think that the searchspace for the rotational movement should be related to the velocity, and not to the current location.

4 Experiments

initial velocity is neglected

in de paper: alpha is 0.1 beta is 4 eta = 0.9

4.1 Original experiments

5 Results

6 Conclusion

a: easy to implement, but with difficulties in finding the right variables to use as there were some inconsistencies in the original paper with variable names.
b: quite original as nothing really looked like it and there were some new ideas, but it wasn't revolutionary since the concept was quite simple and didn't really negate problems with local minimums.

References

1. Abedinia, Oveis, A.N., Ghasemi, A.: A new metaheuristic algorithm based on shark smell optimization. In: Wiley Periodicals. vol. 21, No. 5, pp. 97–116 (2014)

A Table of user defined constants

Variable	Minimum value	Maximum value
M	1	undefined
N	1	undefined
k_{max}	1	undefined
O	0	undefined
α_k	0	1
β_k	0	undefined
η_k	0	1

B Supplementary materials

Next to the report, you are also required to submit the following materials:

- The C++ code of your algorithm. Use sufficient comments, modular coding style and clear variable names. This should only be one file which can directly be plugged into the IOHexperimenter (name the file *GroupXX.cpp*). Make sure this file compiles and runs without errors on linux (the machines in rooms 302-306).
  Tip: You can sse https://git.liacs.nl to host and share code with your teammates. You can log in using your ULCN username and password.
- The runtime data generated by your code. This should be a single zip-file, containing only the data from your algorithm (set algorithm name property to *GroupXX-Animal-Name*).
- Per team member: In an email (to `d.l.vermetten@liacs.leidenuniv.nl`, subject line should include "[NaCo Assignment Review]" + your group number): a grade for your teammates based on their contribution, with some motivation.
- This report in pdf format + the files used to generate it (tex, bib, figures).