A New Approach to Global Optimization: Sheep Optimization

Robert Sawko and Grzegorz Skorupa[1]

[1] Wroclaw Univeristy of Technology, Institute of Information Science and Engineering, Wroclaw, Poland,
email: robert.sawko@student.pwr.wroc.pl, grzegorz.skorupa@student.pwr.wroc.pl

Abstract. The paper presents a new meta-heuristics for solving continuous optimization problems of finding a global optimum. The algorithm is based on the behavior of a specific animal species. The main inspiration for this method was a flock of sheep, which after consuming the grass in a certain area, starts to search for new sources of food when the local sources are depleted. A special penalty function to enforce that kind of behavior is proposed. The penalty function together with a gradient-based optimization algorithm became a mechanism for avoiding local maximums and for more thorough exploration of the set of feasible solutions. The comparison with the basic genetic algorithm is presented.

1 Introduction

Optimization problems are a crucial element of many branches of engineering and science. Problems of this class can arise in technical and nontechnical applications and include such tasks like enhancing the performance of a device, enhancing the parameters of a material, profit maximising or minimizing the time of a job.

Continuous optimization problems become challenging when the number of extreme points or the number of dimensions grow. The necessity to solve more complex tasks as well as the lack of efficient general methods resulted in the elaborating of numerous heuristics for solving optimization problems such as GeneticAlgorithms (GA) [8] and Evolutionary Algorithms (EA) [1], Ant Colony [3], Particle Swarm (PSO) [6], Tabu Search (TS) [2], Simulated Annealing [7] or even Harmony Search Optimization [5]. Extensions of classical optimization algorithms such Penalty Methods [10] have been also developed.

One of the common features of these methods is the ability to adapt to a specific function by applying certain procedures or tuning the parameters. These mechanisms allow the avoidance of local extreme points or improve already found solutions e.g. GA uses mutation operators for obtaining new solutions and fitness based on the selection operator for the improvement. Optimization algorithms are often characterized by two important features: the ability to explore and to exploit the space of feasible solutions.

All of the mentioned techniques were inspired by some phenomenon observed in nature. In this paper a new technique inspired by the behavior of a flock of sheep is presented. The algorithm belongs to a class of penalty methods. The main goal of the algorithm is to find a global extreme point in the presence of many local extreme points.

The rest of the paper is organised as follows: the definition of the optimization problem and a gradient ascent method is recalled in Section 2, Section 3 describes the inspiration more thoroughly and proposes procedures implementing it. Section 4 is devoted to the algorithm itself and in Section 4 some experimental results and comparisons to GA are presented. Some general remarks are concluded in Section 5.

2 Basic Definition and Algorithm

The optimization problem is the problem of finding an extreme point i.e. minimum or maximum of a function in the space of feasible solutions. In the rest of the paper attention will be focused on maximisation. Formally this problem can be formulated as followed: let F be a quality function to be optimized. Then the solution of the optimization problem is defined as:

$$x^* = \arg\max_{x \in X} F(x) \tag{1}$$

where X is a set of feasible solutions and an element of $x \in X$ is a d-dimensional vector $x = [x^{(1)}, x^{(2)}, \ldots, x^{(d)}]$.

One of the basic local optimization algorithms is gradient ascent. In the next section, this algorithm will be modified to achieve a better global performance. Although sheep optimization algorithm will be based on gradient ascent, the proposed methodology is much more general and can be applied in conjunction with other gradient and non gradient methods. For the maximisation problem the formula of the algorithm is the following:

$$x_{n+1} = x_n + \gamma \nabla F(x)_{x=x_n} \tag{2}$$

where x_n is the solution in the n-th iteration and γ is a constant.

3 Inspirations and Corresponding Mechanisms

A flock of sheep or a swarm of locusts located in a region exploit the available sources of food. After consuming all of the available resources, it moves to another location leaving bare ground behind it. Any region after some time becomes unattractive and a flock moves around, always looking for more fertile lands.

3.1 Gradient Ascent with Implemented Sheep Behavior

Now the sheep behavior will be introduced to gradient ascent method. The idea is to translate a movement of flock members, which are constantly searching for more food, into a movement of a set of points in a space of feasible solutions. The movement will be affected by a modified quality function. This modification considers the already visited locations.

A position of one sheep represents an acceptable solution, i.e. a point from set X. In each iteration the sheep moves around the search space and the amount of food in the sheep neighbourhood is reduced. This reduction is an equivalent of food consumption and is represented by a penalty function imposed on quality function.

The quality function can be treated as a starting food resources in the environment. A sheep move around according its to gradient ascent, hence it will always move in

the direction which promises the biggest food amount improvement. For each iteration, quality function with an imposed penalty is the current food resources of the environment. The penalty does not have to be imposed in each iteration. One can define special condition determining whether to consume food or not. Hence sheep has two states: searching for food and food consumption.

In the outcome every region will become less attractive for the sheep and it will move to another searching for more resources. Of course a sheep has no mechanism to distinguish global and local maximum, but remembering the visited places and their corresponding quality function values (without imposed penalty) will give a good approximation of the global maximum position.

3.2 Penalty Functions

In the case of maximisation problem, the penalty function should reduce the value of the quality function in the region. Let $p \in X$ be a point where the penalty was imposed and the form of the penalty function be defined by:

$$k(x,p) = a\exp(-\frac{\|x-p\|^2}{b}), \quad a>0, b>0 \tag{3}$$

where a and b are constant parameters reflecting the strength and the size of penalty respectively. The penalty function is increasing as we approach to point p and decreases to zero otherwise.

Furthermore for a finite sequence of L points $P = \{p_i\}_{i=1}^{L}$, one can define the total penalty function as a sum of all penalty functions:

$$K(x,P) = \sum_{i=1}^{L} k(x,p_i). \tag{4}$$

The resultant quality function perceived by a sheep can be defined as:

$$G(x,P) = g + \frac{F(x)-g}{1+K(x,P)} \tag{5}$$

where g is so called ground level. Ground level should be a real number such that:

$$\forall x \in X \; F(x) > g. \tag{6}$$

The definition of the penalty and its imposition fulfil four important properties:

- When x is close to any of the points from P then the value of G will be less than F.
- When x is far from every point from P then the $G(x,P) \approx F(x)$.
- Greater values of F result in greater penalties i.e.

$$F(x_1) > F(x_2) \Rightarrow (G(x_1,\{x_1\}) - F(x_1)) > (G(x_2,\{x_2\}) - F(x_2)). \tag{7}$$

- Value of G is always greater than the ground level. Thanks to this property one can avoid so called "digging" - the situation in which a sheep imposes a penalty but does not leave a bounded region.

3.3 Movement of Sheep

As it has been already mentioned, the movement will proceed according to gradient ascent method. For the purpose of movement the gradient of G function will be used instead of F. Assuming that the flock consist of M sheep and $x_{m,n}$ is a position of mth sheep in nth iteration, then position of the same sheep in next iteration is given by:

$$x_{m,n+1} = x_{m,n} + \Delta_{m,n} \tag{8}$$

where

$$\Delta_{m,n} = \gamma \nabla G(x, P)|_{x=x_{m,n}} \tag{9}$$

and γ is a constant. Later instead predefined constant a sequence (γ_n) will be used. The sequence should fulfil two properties:

$$\lim_{n\to\infty} \gamma_n = 0, \tag{10}$$

$$\sum_{n=0}^{\infty} \gamma_n = \infty. \tag{11}$$

In the paper the sequence of the form:

$$\gamma_n = \frac{1}{1+\eta n} \tag{12}$$

where η is a constant real number is used.

3.4 Building the Set of Penalty Points

Now conditions for placing penalty points will be discussed. In a case where there is no condition, sheep will consume food in every location that they visit consequently strongly deforming the observed quality function. To avoid that kind of behaviour penalty point is placed only when the position change is small, that is $\|\Delta_{m,n}\| < \epsilon$, where ϵ is a number close to zero.

3.5 Leaving Maximums Neighbourhood

The gradient based mechanism and the penalty function without an additional mechanism do not ensure leaving the local maximum neighbourhood. The second issue is the emergence of a new extreme points or shifting the existing ones(!). The only outcome of using a proposed penalty function is the decrease of function values. On the other hand, this decrease can impact the gradient mechanism especially where the values are near the ground level, because the function will become "smoother" .

Hence a necessity to introduce another mechanism arises. Its main task is to allow a sheep to leave penalised region faster. The simplest method consists in generating a random vector and move according to it as long as some condition is met (this mechanism replaces the gradient). Other solution (used later in experiments) is to multiply gradient by a predefined constant when the length of the replacement is small:

$$\Delta_{m,n} = h \nabla G(x, P)|_{x=x_{m,n}} \tag{13}$$

with $h > 1$. The sheep behaves this way only when the replacement computed by ordinary gradient is small. It is worth mentioning that in such a case $\|\Delta_{m,n}\|$ is always less than $h\epsilon$.

3.6 Algorithm Pseudo-code

Let us denote $\hat{x}$ as a solution returned by an algorithm. Let M be a number of sheep in a flock. Then the sheep optimization algorithm can be formulated as follows:

1. Initialization
   1) n:=0, $P := \emptyset$
   2) generate starting points of sheep: $x_{1,0}, x_{2,0}, \ldots, x_{M,0}$
   3) set starting indices of γ_{k_m} sequence $k_i := 1$, $i = 1, 2, \ldots, M$
   4) $\hat{x} := x_{1,0}$
   5) $m := 1$ (index of a sheep)
2. $F(x_{m,n}) \geqslant F(\hat{x})$ then $\hat{x} := x_{m,n}$
3. Compute: $\Delta_{m,n} := \gamma_{k_m} \nabla G(x, P)|_{x=x_{m,n}}$
4. If $\Delta_{m,n} < \epsilon$ then go to 5, else go to 8.
5. Increase move $\Delta_{m,n} := h\Delta_{m,n}$
6. $k_m := 1$
7. Add new penalty in a current position $P := P \cup \{x_{m,n}\}$
8. $m := m + 1$, $k_m := k_m + 1$
9. if $m \leqslant M$ then go to 2. (loop for each sheep)
10. $n =: n + 1$, $m := 1$
11. If $n \leqslant N$ then return to 2. (loop for each iteration)
12. return $\hat{x}$ as a result. FINISH.

3.7 Similarities to Other Optimization Methods

This approach is one of the penalty methods used together with multistart algorithms to avoid multiple determinations of local minima [10]. These methods are applied to find a starting point for a local optimization algorithm. The obtained starting point should belong to a region of attraction of a better local extreme point. Repeating this procedure should lead to finding a global extreme point. Hence the aim of using penalty functions is the same but they are used in a different way.

The form of the penalty function in sheep optimization is most similar to the one presented in Filled Function method [4]. The impact of the penalty function in the sheep optimization is strongly local i.e. the effect is negligible in places that are far away from penalty points. Filled Function method on the contrary flattens the function in a far way regions and therefore strongly deforms it.

This approach may seam similar to PSO where the social mechanism is used to avoid local extreme points. However the interaction between flock members is not direct and places already visited by the flock member become less attractive instead of drawing attention of other flock members. On the other hand, the sheep based approach may resemble TS where some solutions become not acceptable. A distinctive feature to TS is that proposed solution introduces levels of attractiveness being in consequence more general mechanism than TS, where the level is plus infinity for unacceptable solutions.

4 Experimental Performance Evaluation

The performance of the proposed algorithm has been tested on two functions from de-Jong's test suit [9] namely General Rastrigin Function and Shekel-Foxholes Function. Two variants of the sheep algorithm were utilised $M = 1$ and $M = 20$.

GA has two tunable parameters: p_c and p_m which are crossover and mutation probabilities. To represent a solution, which is real number vector, a standard binary encoding of real numbers was used. Each coordinate was encoded with 16 bits which gave a precision of the order 10^{-4}. Standard two-point crossover and a roulette wheel for selection are used. Mutation operator is a one known from the literature: a change of each bit with probability p_m. The size of the population was 20.

For each setup 20 numerical experiments were conducted. Parameters of the algorithms were set to equalise the algorithms operating time, hence each algorithm had the same time to find a solution. Obviously such approach is implementation dependent but allows performance comparison. Both algorithms were implemented in MATLAB environment. No built in toolbox or external library was used to minimise the impact of implementation. The experiments were preceded by calibration process. Results presented in tables are an average of: distances of solutions found to optimum and quality loss. Additionally the distance of closest(min) and furthest(max) obtained solutions are also presented.

4.1 General Rastrigin Function

General Rastrigin Function can be described by a following formula:

$$f(x) = -10d - \sum_{i=1}^{d}((x^{(i)})^2 - 10\cos(2\pi x^{(i)})) \tag{14}$$

Two cases: $d = 2$ and $d = 20$ were considered. In both cases the optimal solution and its quality function value are:

$$x^* = \vec{0}, \qquad f(x^*) = 0. \tag{15}$$

and the region of optimization:

$$x \in [-20, 20]^d \tag{16}$$

After calibration process parameters of the sheep algorithm were set for $d = 2$ problem as follows: $a = 0.3$, $b = 0.7$, $\epsilon = 0.01$, $h = 500$, $\eta = 0.1$, $g = -60$. After calibrating GA $p_c = 0.8$, $p_m = 0.2$ were chosen. The results of this experiment are presented in Table 1.

In the case of $d = 20$ the parameters were set to: $a = 1$, $b = 0.55$, $\epsilon = 0.01$, $h = 200$, $\eta = 0.1$, $g = -60$ and the parameters of GA were: $p_c = 0.7$, $p_m = 0.05$. The results are presented in Table 2.

Table 1. Results for General Rastrigin function for $d = 2$

Algorithm	N	$f(x^*) - f(\hat{x})$	$\lvert x^* - \hat{x}\rvert$	$\min \lvert x^* - \hat{x}\rvert$	$\max \lvert x^* - \hat{x}\rvert$
S1	15000	0.043	0.043	0.00002	0.994
S20	1000	0.033	0.011	0.001	0.037
GA	800	1.715	0.883	0.011	1.449

Table 2. Results for General Rastrigin function for $d = 20$

Algorithm	N	$f(x^*) - f(\hat{x})$	$\lvert x^* - \hat{x}\rvert$	$\min \lvert x^* - \hat{x}\rvert$	$\max \lvert x^* - \hat{x}\rvert$
S1	20000	0.85	0.76	0.0005	1.41
S20	1500	66.26	0.96	0.44	1.84
GA	2900	35.84	2.91	0.2	5.05

4.2 Shekel-Foxholes Function

Shekel-Foxholes Function for $d = 2$ is defined by:

$$f(x) = 0.002 + \sum_{i=1}^{25} \frac{1}{i + \sum_{j=1}^{d} (x^{(j)} - a_{ij})^2} \tag{17}$$

where

$$a = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & \dots \\ -32 & -32 & -32 & -32 & -32 & -16 & -16 & -16 & \dots \end{pmatrix}. \tag{18}$$

For a function defined this way the optimal solution and a corresponding value of a quality function are:

$$x^* = [-32, -32]^T, \qquad f(x^*) = 1.0178. \tag{19}$$

with the optimization region:

$$x \in [-35, 0]^d \tag{20}$$

The calibration process resulted in the following parameters for sheep optimization algorithms $a = 0.6$, $b = 0.1$, $\epsilon = 0.1$, $h = 200$, $\eta = 0.1$, $g = -0.2$. The parameters of GA were set to $p_c = 0.7$, $p_m = 0.3$. The Table 3 presents the result of numerical experiments.

Table 3. Results for Shekel-Foxholes function $d = 2$

Algorithm	N	$f(x^*) - f(\hat{x})$	$\lvert x^* - \hat{x}\rvert$	$\min \lvert x^* - \hat{x}\rvert$	$\max \lvert x^* - \hat{x}\rvert$
S1	2500	0.002	0.040	0.004	0.102
S20	130	0.03	0.045	0.001	0.117
GA	800	0.07	0.256	0.058	0.512

Table 4. Results for Shekel-Foxholes function $d = 5$

Algorithm	N	$f(x^*) - f(\hat{x})$	$\lvert x^* - \hat{x}\rvert$	$\min \lvert x^* - \hat{x}\rvert$	$\max \lvert x^* - \hat{x}\rvert$
S1	6600	1.164	0.110	0.037	0.181
S20	300	1.750	0.141	0.029	0.289
GA	800	7.927	4.370	0.258	10.913

The last test was made on five dimensional Shekel-Foxhole's function. The parameters of proposed algorithm were set to: $a = 0.7$, $b = 1.3$, $\epsilon = 0.05$, $h = 200$, $\eta = 0.05$, $g = -0.2$ and the parameters of GA were $p_c = 0.6$, $p_m = 0.2$. Results are shown in Table 4.

Presented experiments show that sheep optimization approach can be competitive to GA. In fact most of the obtained results suggest that sheep algorithm performed better and achieved values one order of magnitude closer to maximum point. On the other hand sheep algorithm is described by much more parameters than its simple GA equivalent

hence it is more difficult to calibrate them. Moreover GA is an inefficient algorithm thus a good result in comparison does not prove the efficiency of the sheep optimization. A more thorough comparison should use one of the more sophisticated versions of GA or other algorithms with proved efficiency e.g. the ones using information about gradient. Also a comparison to penalty methods would be valuable.

The results obtained by S20 were slightly worse than the results of S1. This may suggest that using many sheep simultaneously does not have to improve the performance or quality. However this loss may be caused by a smaller number of steps performed by each sheep.

5 Conclusions

The benefit of using sheep optimization is the ability to avoid any local extreme point. That is why this approach can be useful when dealing with a strongly varying function. Sheep optimization can become the first estimation of a global extreme giving a set of positions which can be further exploited by different algorithms, especially when used as part of a hybrid algorithms

In a future work, we plan to develop a mechanism for leaving the neighbourhood of local optimum. A more thorough comparison with different meta-heuristics is needed. We also plan to use sheep optimization in problems related to game theory, namely searching for the Nash equilibrium.

Bibliography

[1] D. Ashlock. *Evolutionary Algorithms for Modeling and Optimization.* Springer, 2006.

[2] D. Cvijovic and J. Klinowski. Taboo search - an approach to the multiple minima problem. *Science*, 267:664–666, 1995.

[3] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man and Cybernetics Part B*, 26(1):2941, 1996.

[4] R.P. Ge. A filled function method for finding a global minimizer. In *Dundee Biennial Conference on Numerical Analysis*, 1983.

[5] Z.W. Geem, J. H. Kim, and G.V. Loganathan. A new heuristic optimization algorithm: Harmony search. *Simulation*, 76(2):60–68, 2001.

[6] J. Kennedy and R. Eberhart. *Swarm Intelligence.* Morgan Kaufman Publishers, 2001.

[7] S. Kirpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[8] M. Mitchell. *An Introduction to Genetic Algorithms.* The MIT Press, 1998.

[9] D. Nikos. The function testbed. Website, 2007. http://www.it.lut.fi/ip/evo/functions/functions.html.

[10] A. Torn and A. Zilinskas. *Global optimization.* Springer-Verlag New York, Inc., New York, NY, USA, 1989.