Mutation-Based Artificial Fish Swarm Algorithm for Bound Constrained Global Optimization

Ana Maria A.C. Rocha* and Edite M.G.P. Fernandes†

*Department of Production and Systems, University of Minho, 4710-057 Braga, Portugal
†Algorithm R&D Center, University of Minho, 4710-057 Braga, Portugal

Abstract. The herein presented mutation-based artificial fish swarm (AFS) algorithm includes mutation operators to prevent the algorithm to falling into local solutions, diversifying the search, and to accelerate convergence to the global optima. Three mutation strategies are introduced into the AFS algorithm to define the trial points that emerge from random, leaping and searching behaviors. Computational results show that the new algorithm outperforms other well-known global stochastic solution methods.

Keywords: Global optimization, Artificial Fish Swarm, Mutation
PACS: 02.60.Pn

INTRODUCTION

This paper aims at introducing an artificial fish swarm (AFS) algorithm that relies on mutation operators to describe the fish behaviors that mostly contribute to diversifying in the search space: random, searching and leaping behaviors. The AFS algorithm is a recent and easy to implement artificial life computing algorithm that generates a sequence of approximations to the solution of a global optimization problem, while simulating fish swarm behaviors inside water. It uses a population of points to identify promising regions looking for a global solution [4, 6, 9]. Fishes desire to stay close to the swarm, avoiding collisions within the group, protecting themselves from predators and looking for food. The five fish behaviors inside water are:

i) random behavior - fish swims randomly in water looking for food and other companions;
ii) searching behavior - fish tends directly and quickly to regions with more food, by vision or sense;
iii) swarming behavior - fish naturally assembles in groups which is a living habit in order to guarantee the existence of the swarm and avoid dangers;
iv) chasing behavior - fish finds the food dangling quickly after a fish, or a group of fishes, in the swarm that discovered food;
v) leaping behavior - fish leaps to look for food in other regions when it stagnates in a region.

Here, we are specially interested in solving bound constrained problems of the form:

$$\text{minimize}_{x\in\Omega}\, f(x) \tag{1}$$

where $f:\mathbb{R}^n\to\mathbb{R}$ is a nonlinear function and $\Omega=\{x\in\mathbb{R}^n : l\le x\le u\}$ is the feasible region. The objective function f may be non-smooth and may possess many local minima in the search space Ω, since we do not assume that f is convex. Many derivative-free algorithms and heuristics have been proposed to solve (1), namely those based on swarm intelligence. Probably the most well-known are the particle swarm optimization [1], the ant colony [7] and the artificial bee colony [5] algorithms. The most important global search techniques invoke exploration and exploitation search procedures aiming at:

i) diversifying the search in all the search space;
ii) intensifying the search in promising areas of the search space.

An efficient AFS algorithm should be capable of exploring the whole search space as well as exploiting around the neighborhood of a reference point, for example, the best point of the population. Previous experiments have shown that AFS algorithms may be trapped into local optimum, although the leaping behavior has been helping to jump out

from local solutions. Furthermore, in a population-based method, it is very important to obtain the optimal solution in a minimum time period. This paper proposes the introduction of some mutation operators into the AFS algorithm in order to diversify the search and prevent the algorithm from falling into a local optimum. The remaining paper is organized as follows. Firstly, the main steps of the AFS algorithm will be described and the proposed mutation-based AFS algorithm will be introduced. Finally, the numerical results and remarks will be shown.

MUTATION-BASED AFS ALGORITHM FOR BOUND CONSTRAINED PROBLEMS

This is a list of the used notation: the position of an artificial fish in the search space is herein denoted by a point, $x^i \in \mathbb{R}^n$, the ith point of a population. x^{best} is the point that has the least objective function value in Ω; $x^i_j \in \mathbb{R}$ is the jth ($j = 1, \ldots, n$) component of the point x^i of the population and p_{size} is the number of points in the population. Hereafter PI represents the set of indices $\{1, 2, \ldots, p_{\text{size}}\}$.

The initial population is generated randomly and should cover the entire search space Ω. Fish/point movements are defined according to its 'visual scope'. This is the closed neighborhood centered at x^i with radius $v > 0$ defined by $v = \delta \max_{j \in \{1,\ldots,n\}} (u_j - l_j)$, where δ is a positive visual parameter. It has been shown that a reduction of δ accelerates the convergence to the solution [6]. Hence, every iteration, the visual parameter is updated using $\delta = \max\{\delta_{\min}, \mu_\delta \delta\}$, where $0 < \mu_\delta < 1$ and $\delta_{\min} > 0$. Let I^i be the set of indices of the points inside the 'visual scope' of x^i ($i \notin \mathrm{I}^i$ and $\mathrm{I}^i \subset \mathrm{PI}$) and np^i the number of points in the 'visual scope'. If the condition $\mathrm{np}^i / p_{\text{size}} \leq \theta$ holds, where $\theta \in (0, 1]$ is the crowd parameter, then the 'visual scope' of x^i is said to be not crowded. Depending on the relative positions of the points in the population, three possible situations may occur:

- when $\mathrm{np}^i = 0$, the 'visual scope' is empty, and the point x^i, with no other points in its neighborhood to follow, has a random behavior moving randomly inside the 'visual scope';
- when the 'visual scope' is crowded, the point has some difficulty in following any particular point, and has a searching behavior choosing randomly another point (from the 'visual scope') and moving towards it;
- when the 'visual scope' is not crowded, the point is able either to swarm moving towards the central point of the 'visual scope' (swarming behavior), or to chase moving towards the best point of the 'visual scope' (chasing behavior). The original AFS algorithm simulates both movements and chooses the best in the sense that a better objective function value is obtained.

Details concerning the referred behaviors follow. The swarming behavior is characterized by a movement towards the central point of the 'visual scope' of x^i, $c = \sum_{j \in \mathrm{I}^i} (x^j / \mathrm{np}^i)$. However, the swarming behavior is activated only if the central point has a better objective function value than that of x^i. Otherwise, the point x^i follows the searching behavior. In the searching behavior, a point is randomly chosen inside the 'visual scope', x^{rand}, and a movement towards it is carried out if the random point improves over x^i. Otherwise, the point has a random behavior. The chasing behavior is carried out when a point, denoted by $x^{\min}$, with the minimum objective function value inside the 'visual scope' of x^i, satisfies $f(x^{\min}) \equiv \min\{ f(x^j) : j \in \mathrm{I}^i \} < f(x^i)$. However, if this last condition is not satisfied then the point activates the searching behavior. Finally, when the best point of the population, x^{best}, does not change for a certain number of iterations, the algorithm may have fallen into a local minimum. To be able to overcome this 'stagnation' and try to converge to the global minimum, the leaping behavior is implemented. At every p_{size} iterations, a point is randomly selected from the population and a random movement is carried inside the set Ω [9].

The modifications that we have introduced in the past into the AFS algorithm to improve convergence to global solutions are now described. Firstly, each point movement, either towards the central c, the $x^{\min}$ or x^{rand}, defines a trial point, herein denoted by y^i, that is defined componentwise by $y^i_j = x^i_j + \xi d^i_j$, $j = 1, \ldots, n$, where d^i represents the vector with the direction of movement and ξ is a uniformly distributed value in $[0, 1]$. Then any point's component outside the bounds, is projected onto Ω. At the end of each iteration, the algorithm implements a selection operation aiming at accepting the trial point y^i only if it improves over the current point x^i. After defining the population for the next iteration and detecting the best point of the population, a local search procedure is used to refine locally the search around x^{best}. We propose the well-known Hooke and Jeeves pattern search algorithm [3].

Hooke and Jeeves local search: This is a derivative-free deterministic method that exploits the neighborhood of a point for a better approximation using two types of moves: the exploratory move and the pattern move. It is a variant of the well-known coordinate search method (a search along the coordinate axes). It incorporates a pattern move to accelerate the progress of the algorithm, by exploiting information obtained from the search in previous successful

iterations. At each iteration the exploratory move carries out a coordinate search around the best point, with a step length Δ. If a new trial point, y, with a better function value than x^{best} is encountered, the iteration is successful and Δ is maintained. Otherwise, the iteration is unsuccessful and Δ should be reduced. If the previous iteration was successful, the vector $y - x^{\text{best}}$ defines a promising direction and a pattern move is then implemented, which means that the exploratory move is carried out around the trial point $y + (y - x^{\text{best}})$, rather than the current point y. Then, if the coordinate search is successful, the returned point is accepted as the new point; otherwise, the pattern move is rejected and the method reduces to a coordinate search around y. We refer to [3] for details.

When the 'visual scope' of a point x^i is not crowded, the original AFS algorithm tries the swarming and the chasing behaviors to check which one is the best. Function values at $x^{\min}$ and c have to be evaluated and compared to each other, as well as with $f(x^i)$. To reduce function evaluations, the priority-based strategy is proposed in [6].

Priority-based strategy: In order to accelerate convergence, a priority-based strategy has been implemented only when the 'visual scope' of a point x^i is not crowded [6]. It simulates one behavior at each time instead of trying both swarming and chasing behaviors at the same time. The chasing behavior is ranked with highest priority, so that the movement in direction to $x^{\min}$ is carried out first if $f(x^{\min}) < f(x^i)$. Otherwise, the swarming behavior will be the alternative. So, the movement in direction to c is then carried out if $f(c) < f(x^i)$. However, if the latter condition does not hold then the point has a searching behavior [6].

We now introduce the main features of the new mutation-based AFS algorithm. Mutation operators are implemented ir order to describe the random, searching and leaping behaviors.

Mutation operators: Differential evolution (DE) is an evolutionary algorithm proposed in [8] for bound constrained global optimization. It generates new trial points by combining a current point and several other points of the same population. DE follows three operations to generate the population for the next iteration. One is the mutation and it aims to create the therein called mutant points v^i, $i = 1, \ldots, p_{\text{size}}$, relative to the current points x^i. The most used mutation strategy is referred as DE/rand/1, but there are others available in the literature:

$$\begin{array}{llll} \text{DE/rand/1} & v^i = x^{r_1} + F_1\,(x^{r_2} - x^{r_3}) & \text{DE/target-to-rand/1} & v^i = x^i + F_2\,(x^{r_1} - x^i) + F_1(x^{r_2} - x^{r_3}) \\ \text{DE/best/1} & v^i = x^{\text{best}} + F_1\,(x^{r_1} - x^{r_2}) & \text{DE/target-to-best/1} & v^i = x^i + F_1\,(x^{\text{best}} - x^i + x^{r_1} - x^{r_2}) \end{array} \tag{2}$$

where indices r_j are randomly chosen from the set PI, mutually different and different from the running index i. F_1 and F_2 are real and constant parameters from $[0,2]$ and $[0,1]$ respectively. The mutation operation plays a vital role in the DE algorithm in order to explore the entire search space and, at the same time, to expedite convergence. For example, DE/rand/1 mutation strategy explores the entire search space but desaccelerates convergence. On the other hand, DE/best/1 strategy exploits around the best point found so far and accelerates convergence. With this mutation a local solution may be obtained before the global solution can be reached. Hence, it is very important to balance both random movements. Motivated by the success of mutation operators in DE, we select three strategies to define the trial points, y^i, that emerge from random, leaping and searching behaviors, as follows:

$$\begin{array}{rl} \text{for the random behavior:} & y^i = x^i + F_2\,(x^{r_1} - x^i) + F_1(x^{r_2} - x^{r_3}) \\ \text{for the leaping behavior:} & y^i = x^{r_1} + F_1\,(x^{r_2} - x^{r_3}) \\ \text{for the searching behavior:} & y^i = x^i + F_1\,(x^{\text{best}} - x^i + x^{r_1} - x^{r_2}) \end{array} \tag{3}$$

where r_j, $j = 1,2,3$ are defined as previously described.

NUMERICAL RESULTS AND REMARKS

For a practical assessment of the proposed modifications, numerical experiments are carried out involving nine standard benchmark test problems. The algorithms are coded in C and the results were obtained in a PC with a 3GHz Pentium IV microprocessor and 1Gb of memory. Each problem was solved 30 times and a population of $p_{\text{size}} = \min\{100, 10n\}$ points is used. Other parameters are set as follows: initial δ is 1, $\delta_{\min} = 0.1$, $\mu_\delta = 0.9$, $\theta = 0.8$, $V = p_{\text{size}}$, $r = n$, $F_1 = 0.5$, $F_2 = 1$. First, we compare the results of a previous AFS algorithm [6], 'AFS', of the herein proposed mutation-based AFS algorithm, '**m**-AFS', with the electromagnetism-like mechanism, 'EM', of [2], using the following condition to judge the success of the run

$$\left| f(x^{\text{best}}) - f_{\text{opt}} \right| \leq 0.0001 \left| f_{\text{opt}} \right| \quad \text{OR} \quad n f_{\text{eval}} > 20000, \tag{4}$$

where $f(x^{\text{best}})$ is the best solution found so far, f_{opt} is the known optimal solution, and nf_{eval} is the number of function evaluations required to obtain a solution with the specified accuracy. For each method, Table 1 contains the registered average number of function evaluations obtained in the 30 runs. In the table, 'Prob.' lists the acronym of the tested problems.

To compare the performance of our AFS algorithms with that of improved particle swarm optimization (PSO) algorithms - variants 'PSO-RPB' and 'PSO-HS' - proposed in [1], as well as with the natural competitor, the differential evolution 'DE' (presented in [1]), we solved again the problems using the stopping condition therein selected [1]:

$$\left| f(x^{\text{best}}) - f_{\text{opt}} \right| \leq 0.001 \quad \text{OR} \quad nf_{\text{eval}} > 20000. \tag{5}$$

The last five columns of Table 1 show the average number of function evaluations reached by AFS, **m**-AFS, PSO-RPB, PSO-HS and DE. From the comparison we may conclude that the mutation-based AFS algorithm is better than the previous AFS algorithm in terms of efficiency (average number of function evaluations). Further, mutation-based AFS algorithm is considered rather competitive when compared with the EM algorithm. When we consider the second set of experiments using condition (5), we observe that both AFS and **m**-AFS improve over both variants of PSO in comparison and DE.

TABLE 1. Results of average number of function evaluations.

Prob.	f_{opt}	stopping condition in (4)			stopping condition in (5)				
		AFS	**m**-AFS	EM	AFS	**m**-AFS	PSO-RPB	PSO-HS	DE
BR	0.39789	550	475	315	651	438	2652	2018	1305
CB6	-1.0316	331	247	233	246	245	2561	2390	1127
GP	3.00000	676	417	420	562	485	2817	1698	884
H3	-3.86278	2930	1891	1114	1573	1142	3564	2948	1238
H6	-3.32237	7091	2580	2341	7861	2845	8420	8675	7053
S5	-10.1532	3928	1183	3368	3773	1150	6641	6030	5824
S7	-10.4029	4033	1103	1782	2761	1240	6860	6078	5346
S10	-10.5364	2069	1586	5620	2721	1190	6747	5602	4822
SBT	-186.731	472	523	358	659	516	4206	6216	2430

The goal of this research is to improve the performance of the AFS algorithm, preventing convergence to local solutions and accelerating convergence to a global solution. The results that come out from the introduction of various mutation operators, as presented in DE-type methods, to translate AFS behaviors, like searching, random and leaping, show the goodness of the proposal. Future developments will focus on an adaptive strategy to define parameters F_1 and F_2.

REFERENCES

1. M.M. Ali, P. Kaelo, *Improved particle swarm algorithms for global optimization*, Applied Mathematics and Computation, 196 (2008) 578–593.
2. S.I. Birbil, S. Fang, *An electromagnetism-like mechanism for global optimization*, Journal of Global Optimization, 25 (2003) 263–282.
3. R. Hooke, T.A. Jeeves, *Direct search solution of numerical and statistical problems*, Journal of the Association for Computing Machinery, 8 (1961) 212–229.
4. M. Jiang, Y. Wang, S. Pfletschinger, M.A. Lagunas, D. Yuan, *Optimal multiuser detection with artificial fish swarm algorithm*, In: D.-S. Huang, L. Heutte, M. Loog (eds.), CCIS 2, ICIC 2007, Springer-Verlag, 1084–1093 (2007).
5. D. Karaboga, B. Basturk, *A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm*, Journal of Global Optimization, 39 (2007) 459–471.
6. A.M.A.C. Rocha, E.M.G.P. Fernandes, T.F.M.C. Martins, *Novel fish swarm heuristics for bound constrained global optimization problems*, Lecture Notes in Computer Science, Vol 6784, Part III, B. Murgante, O. Gervasi, A. Iglesias, D. Taniar, B. Apduhan (Eds.) 185–199 (2011).
7. K. Socha, M. Dorigo, *Ant colony optimization for continuous domains*, European Journal of Operational Research, 185 (2008) 1155–1173.
8. R. Storn, K. Price, *Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces*, Journal of Global Optimization, 11 (1997) 341–359.
9. X. Wang, N. Gao, S. Cai, M. Huang, *An artificial fish swarm algorithm based and ABC supported QoS unicast routing scheme in NGI*, In: G. Min et al. (eds.) ISPA 2006, LNCS, vol. 4331, 205–214 (2006).