

Embedding a Competitive Ranking Method in the Artificial Fish Swarm Algorithm for Global Optimization

Ana Maria A.C. Rocha¹ Edite M.G.P. Fernandes², Joana P. Fernandes³ & Tiago F.M.C. Martins⁴

Department of Production and Systems, School of Engineering, University of Minho, 4710-057 Braga, Portugal

¹arocha@dps.uminho.pt; ²emgpf@dps.uminho.pt

Algorithm R & D Center, Portugal

³joanapfernandes@gmail.com; ⁴martins.tiago41@gmail.com

Abstract

Nonlinear programming problems are known to be difficult to solve, especially those that involve a multimodal objective function and/or non-convex and at the same time disjointed solution space. Heuristic methods that do not require derivative calculations have been used to solve this type of constrained problems. The most used constraint-handling technique has been the penalty method. This method converts the constrained optimization problem to a sequence of unconstrained problems by adding, to the objective function, terms that penalize constraint violation. The selection of the appropriate penalty parameter value is the main difficulty with this type of method. To address this issue, we use a global competitive ranking method. This method is embedded in a stochastic population based technique known as the artificial fish swarm (AFS) algorithm. The AFS search for better points is mainly based on four simulated movements: chasing, swarming, searching, and random. For each point, the movement that gives the best position is chosen. To assess the quality of each point in the population, the competitive ranking method is used to rank the points with respect to objective function and constraint violation independently. When points have equal constraint violations then the objective function values are used to define their relative fitness. The AFS algorithm also relies on a very simple and random local search to refine the search towards the global optimal solution in the solution space. A benchmarking set of global problems is used to assess this AFS algorithm performance.

Keywords: Global optimization, Artificial Fish Swarm, Constraint fitness ranking.

1. Introduction

The algorithm herein presented is a stochastic optimization method, called Artificial Fish Swarm (AFS) algorithm, with a constraint-handling method based on a competitive ranking of points with respect to the objective function and constraint violation independently, to solve constrained global optimization problems.

A variety of constraint-handling methods have been developed in the last decades. The most common approach to handle constraints uses a penalty function. The history of penalty functions began with the sequential unconstrained minimization technique by Fiacco and McCormick [5] in which the constrained problem is solved by a sequence of unconstrained optimization problems. There are many types of penalty functions used in optimization: static penalty where the penalty parameter is fixed [6]; dynamic penalty where the penalty factors are derived from the current iteration counter [9]; annealing penalty where the penalty is increased over time in a Simulated Annealing [10] manner; adaptive penalty where the penalty is scaled based on the success or failure of the search [19]; and death penalty where the solution is assigned infinite fitness if it violates any of the constraints [12]. The penalty function approach although simple, requires assignment of penalty factors which are often obtained based on trial and error and the result of optimization is known to be highly sensitive to the choice of the penalty factors. Runarsson and Yao [15] have proposed stochastic ranking to balance between the objective function and the constraint violation. Although this approach does not need any penalty factors, it uses a probability value (between 0.4 and 0.5) to compare infeasible individuals.

Another method for constraint-handling is to handle feasible and infeasible solutions separately. Deb [2] implemented three feasibility dominance rules where the objective value is used as the fitness for

the feasible solutions and the total constraint violation as fitness for infeasible solutions. They can be described as follows: 1) given two feasible points, pick the one with better objective function value; 2) if both points are infeasible, pick the point with lower constraint violation; 3) given a pair of feasible and infeasible points, pick the feasible one.

Dominance based constraint-handling approaches convert the constrained optimization problem (with k objectives) into a multi-objective ($k + 1$ objectives or $k + m$ objectives, where m is the number of constraints) unconstrained optimization problem [17].

In this paper a constraint-handling approach that uses a competitive ranking of the objective values and the constraint violations separately is embedded in the AFS algorithm. A fitness function based on this competitive ranking framework will be introduced to compare points in the same population. To select points between different populations a feasibility dominance criterion is introduced.

The remainder of this paper is organized as follows. Section 2 describes the proposed AFS algorithm and its properties and Section 3 presents the main ideas concerned with the constraint-handling approach, namely the problem definition, the competitive ranking framework and the feasibility dominance criterion. In Section 4 we report our numerical experiments, including a comparison between four fitness function's variants. Finally, Section 5 contains the conclusions and ideas for future work.

2. Artificial Fish Swarm Algorithm

In this section we present a stochastic population-based algorithm that simulates fish swarm behaviors in water. This is an artificial life computing algorithm that has been used in some engineering context [7, 8, 18, 20]. We will use the words 'fish' and 'point' interchangeably throughout the paper. The artificial fish swarm algorithm is based on swarm intelligence and uses a population (or swarm) of points to identify promising regions looking for a global solution.

The artificial fish is a fictitious entity of a true fish. The AFS movements are simulations and interpretations of the below listed fish behaviors [7]:

- i) *random* behavior - in general, fish swims randomly in water looking for food and other companions;
- ii) *searching* behavior - this is a basic biological behavior since fish tends to the food; when fish discovers a region with more food, by vision or sense, it will go directly and quickly to that region;
- iii) *swarming* behavior - when swimming, fish will naturally assemble in groups which is a living habit in order to guarantee the existence of the swarm and avoid dangers;
- iv) *chasing* behavior - when a fish, or a group of fishes, in the swarm discovers food, the others in the neighborhood will find the food dangling quickly after it.

The specific and used notation in the AFS algorithm is as follows: n represents the number of variables, $x^i \in \mathbb{R}^n$ denotes the i th point of the population; $x_j^i \in \mathbb{R}$ is the j th ($j = 1, \dots, n$) coordinate of the point x^i ; and p_{size} is the number of points in the population. We now present the main movements of the points inside the population. In the remaining part of this section, we assume that the variables are subject to simple bounds, i.e., $l \leq x \leq u$, where $l, u \in \mathbb{R}^n$. The term food in the fish swarm system corresponds to a minimum in the optimization context.

A crucial parameter of the artificial fish swarm algorithm is a positive constant v that represents the ray of a closed neighborhood of x^i – the 'visual scope' – herein defined by

$$v = \delta \max_{j \in \{1, \dots, n\}} (u_j - l_j), \quad (1)$$

where δ is a positive visual parameter that is reduced over the iterative process using the update formula $\delta = \max\{\delta_{\min}, \kappa_\delta \delta\}$, with $0 < \kappa_\delta < 1$, and $\delta_{\min} > 0$. The set of indices of the points inside the 'visual scope' of point x^i is denoted by I^i , where $i \notin I^i$ and $I^i \subset \{1, \dots, p_{size}\}$, and np^i is the number of points inside the 'visual scope'. Depending on the relative positions of the points inside the visual, the potential movements to define trial points from the current point x^i are:

- i) when $np^i = 0$, the 'visual scope' is empty, and the point x^i , with no other points in its neighborhood to follow, moves randomly searching for a better region;
- ii) when the 'visual scope' is crowded, the point has some difficulty in following any particular point, and the searching behavior is simulated; the point searches for a better region choosing randomly another point (from the 'visual scope') and moves towards it;

- iii) when the ‘visual scope’ is not crowded, the point is able either to swarm moving towards the central or to chase moving towards the best point, inside the ‘visual scope’.

The condition that decides when the ‘visual scope’ of x^i is not crowded is

$$\frac{np^i}{p_{size}} \leq \theta, \quad (2)$$

where $\theta \in (0, 1]$ is the crowd parameter. In this situation, point x^i has the ability to chase or to swarm.

The chasing behavior is carried out if the movement towards the best point inside the ‘visual scope’ of x^i , herein denoted by x^{\min} , improves over x^i ; otherwise, the searching behavior is activated.

The swarming behavior is characterized by a movement towards the central point in the ‘visual scope’ of x^i , defined by

$$c = \frac{\sum_{j \in I^i} x^j}{np^i}. \quad (3)$$

However, this movement is carried out only if the central point improves over x^i ; otherwise, the point x^i follows a searching behavior. We refer to [4, 14, 18, 20] for further details.

We remark that each new point computed as described above is only a trial point to the next iteration. In fact, the AFS algorithm includes a selection procedure aiming to accept trial points only if they improve over the previous one.

There is also a local procedure in the AFS algorithm, aiming to gather the local information around the best point of the population. It corresponds to a simple random line search applied coordinate by coordinate to one point only - the best point of the population, denoted by x^{best} . The main steps are as follows. For each coordinate j ($j = 1, \dots, n$), x^{best} is assigned to a trial point y . Next, a random movement of length

$$\nu \max_j (u_j - l_j), \quad \nu > 0$$

is carried out and if a better point is obtained within \max_{local} iterations, x^{best} is replaced by y , the search ends for that coordinate and proceeds to the next coordinate.

3. Constraint-handling approach

This section describes the constraint-handling approach. First, we display the definition of the constrained optimization problem; then the competitive ranking method is exposed, and the feasibility dominance criterion used to assess the relative fitness of two points from different populations is presented.

3.1. Problem definition

The general form of the problem of finding a global solution of a nonlinear constrained optimization problem herein considered is:

$$\begin{aligned} & \text{minimize} && f(x) \\ & \text{subject to} && g_i(x) \leq 0, i = 1, \dots, p \\ & && h_j(x) = 0, j = 1, \dots, m \\ & && x \in \Omega \end{aligned} \quad (4)$$

where at least one of the functions $f: \mathbb{R}^n \rightarrow \mathbb{R}$, $g_i: \mathbb{R}^n \rightarrow \mathbb{R}$, $i = 1, \dots, p$ and $h_j: \mathbb{R}^n \rightarrow \mathbb{R}$, $j = 1, \dots, m$ is a nonlinear function and $\Omega = \{x \in \mathbb{R}^n : l \leq x \leq u\}$. The objective function f may be non-smooth and may possess many local minima in the feasible region since we do not assume that f is convex.

The constraint violation of a point x is measured by

$$viol(x) = \sum_{i=1}^p \max\{0, g_i(x)\} + \sum_{j=1}^m |h_j(x)|. \quad (5)$$

In the sequel, a point x with $viol(x) = 0$ is feasible, whereas if $viol(x) > 0$ then the point is infeasible.

3.2. Global competitive ranking scheme

In this section the ideas of the constraint-handling technique based on the global competitive ranking scheme are presented. This is a novel method of ranking points in a population, in order to strike the

right balance between objective and penalty functions [16]. The comparison criterion may be based either on the objective function or the constraint violation, a choice randomly determined by a user-specified probability term P_f . In this method, a point is ranked by comparing it against all other points of the population. We want to remark that this is different from the stochastic ranking approach from Runarsson and Yao [15] where only adjacent points compete for a given rank. In the global competitive ranking scheme, special consideration is given to tied ranks, where in this case the same lower rank will be used. In the competitive ranking scheme, it is assumed that either the objective or the constraint violation will be used in deciding a point's rank.

First we rank the points with respect to objective function value, r_1 , and then with respect to constraint violation, r_2 , independently. All the points in the population are sorted in ascending order based on the objective function value to give r_1 . Then, r_2 is determined when all the points in the population are sorted in ascending order based on the value of the constraint violation. Table 1 shows an example of this ranking assignment for a population with six points.

Table 1: Example of ranking assignment for a population with six points

Point	$f(x)$	$viol(x)$	r_1	r_2
x^1	4	1.20	2	6
x^2	5	0.00	4	1
x^3	4	0.80	2	4
x^4	2	0.73	1	2
x^5	19	0.73	6	2
x^6	10	1.15	5	5

Clearly, the ranking vectors r_1 and r_2 are of the same order of magnitude and each one ranges from 1 to p_{size} (at maximum), where p_{size} is the size of the population. As a result, these vectors can be easily manipulated without bias.

3.3. Fitness functions

In Runarsson and Yao [16], the fitness function for the minimization process is given by the probability that point x^i holds its rank when challenged by any other point of the population:

$$\Phi_1(x^i) = P_f \frac{r_1(i) - 1}{p_{size} - 1} + (1 - P_f) \frac{r_2(i) - 1}{p_{size} - 1}, \quad (6)$$

where the permutations $r_1(i)$ and $r_2(i)$ correspond to the ranking of point x^i based on the objective and constraint violation, respectively, in a population of p_{size} points. P_f indicates the probability that a comparison is done based on the objective function only.

In practice, the probability should take a value $0 < P_f < 0.5$ in order to guarantee that a feasible solution may be found. The closer the probability is to 0.5, the greater the emphasis will be on minimizing the objective function. Previous studies [15] found that a value of $P_f = 0.45$ is often sufficient to establish a pressure against infeasible solutions. When $P_f = 0$ the ranking is equivalent to an over-penalization. Hence, the strength of the pressure can be adjusted easily by adjusting P_f .

In our approach, the probability term P_f varies in a stochastic way, i.e., we use a random number uniformly distributed in $[0, 1]$, λ , so that the establishing pressure against either feasible or infeasible solutions varies randomly. Thus, the herein proposed fitness function for the minimization problem becomes

$$\Phi_2(x^i) = \lambda \frac{r_1(i) - 1}{p_{size} - 1} + (1 - \lambda) \frac{r_2(i) - 1}{p_{size} - 1}. \quad (7)$$

Besides, to further explore the search space for points with better objective function value, and if the point x^i is feasible, the algorithm sets $\lambda = 1$.

When solving some difficult problems an over-penalization is occasionally required. A rather simple strategy consists of extending the fitness function in Eq. (7), replacing $(1 - \lambda)$ by $(p + m) - \lambda$ as shown below:

$$\Phi_3(x^i) = \lambda \frac{r_1(i) - 1}{p_{size} - 1} + (p + m - \lambda) \frac{r_2(i) - 1}{p_{size} - 1} \quad (8)$$

only when x^i is infeasible. This over-penalization aims to drive the infeasible points into the feasible region since when assessing their relative fitness they lose over the feasible points in comparison.

Ho and Shimizu's paper [11] propose an addition of ranking method which is able to balance the objective function against the constraint violation without requiring any additional parameters, where the fitness function is also based on the competitive ranking r_1 and r_2 , and is given by

$$\Phi_4(x^i) = r_1(i) + r_2(i). \quad (9)$$

Here, the fitness function resumes to $\Phi_4(x^i) = r_1(i)$ for feasible points. In Eq. (9), the $r_1(i)$ and $r_2(i)$ terms serve as the penalty function to penalize the infeasible solutions. On the other hand, the $r_1(i)$ term enables us to relate the infeasible individuals to the feasible individuals based on the $f(x^i)$ value alone. The consideration of this term allows one to retain those infeasible solutions with only slight constraint violation and a small objective function value to the next iteration efficiently. This retainment is necessary to maintain the diversity of the population by exploring into the infeasible regions of the search space. Table 1, for example, shows that point x^4 ($\Phi_4(x^4) = 3$), although infeasible, is preferred over the feasible point x^2 because the latter has a larger $f(x^2)$ value ($\Phi_4(x^2) = 5$).

Finally note that, the fitness function is used to assess each point of the population and also used to compare points in the same population. Namely, it is used to determine the point with better fitness function value, x^{best} , the point with worst fitness function value, x^{worst} , and x^{min} . This latter is computed when the chasing behavior is activated.

3.4. Feasibility dominance criterion

The previous ranking method should not be used when comparing a point from the current population with a trial point - a potential point for the next population -, unless the competitive ranking process is repeated with all the involved points. A simpler alternative when a point from the current population, z , is to be compared with another point that is not in the current population, w , is to implement the feasibility dominance criterion. This criterion picks z against w if

$$viol(z) < viol(w) \text{ or } (viol(z) = viol(w) \text{ and } f(z) < f(w)).$$

In the proposed algorithm, this feasibility dominance criterion is implemented:

- i) during the selection procedure, that aims to accept a trial point only if it improves over the point of the current population;
- ii) when the swarming behavior is simulated, i.e., when a movement towards the central point inside the 'visual scope' is tried;
- iii) during the local procedure, which aims to gather the local information around the best point of the population, and defines a new trial point that will be accepted if it improves over x^{best} according to this feasibility dominance criterion.

3.4. Stopping condition

The algorithm terminates when the following condition is verified:

$$(viol(x^{best}) \leq 10^{-3}\epsilon \text{ and } |f^* - f(x^{best})| \leq \epsilon |f(x^{best})|) \text{ or } nit \geq n_{max\ it} \quad (10)$$

where f^* represents the known global optimal solution, $f(x^{best})$ is the objective function value of the best point of the population, nit denotes the iteration counter and $n_{max\ it}$ is the maximum number of iterations allowed. The value of $viol(x)$ represents a measure of constraint violation as previously defined in Eq. (5).

4. Numerical results

In this section, we report the results of our numerical study, after running a set of 24 benchmark constrained global problems, described in full detail in [13]. The problems are known as g01-g24 (the 'g' suit, where six problems have only equality constraints, thirteen have inequality constraints, five have both equalities and inequalities and all have simple bounds). Not all problems have multi-modal objective functions, although some are difficult to solve. The best known solution for problem g20 is slightly

infeasible. We remark that g02, g03, g08 and g12 are maximization problems that were transformed and solved as minimization ones. The algorithm is coded in C# programming language, and the results were obtained in a computer Intel(R) Core(TM)2 Duo CPU 2.93 GHz, with 2.92GHz and 2.97 GB of RAM, running Microsoft Windows XP V2002.

Since the algorithm relies on some random parameters and variables, we solve each problem 30 times and take average of the obtained solutions, herein denoted by f_{avg} . The best of the solutions found after all runs is denoted by f_{best} . The size of the population depends on n , and since some problems have large dimension, $n > 20$, we choose $p_{size} = \min\{200, 10n\}$. Some of the fixed AFS parameters are set in this study as follows: the initial $\delta = 1$, $\kappa_\delta = 0.9$, $\delta_{\min} = 10^{-8}$ and $\theta = 0.8$ as previously used in [14]. In the local search, we use the values $\nu = 0.001$ and $\max_{local} = 10$ since good accuracy solutions were obtained at a reasonable computational cost. The parameters used in the stopping condition Eq. (10) are: $\epsilon = 10^{-4}$ and $n_{max\ it} = 1500$.

4.1. Performance profiles

To compare the performance of the four fitness functions, we use the performance profiles as described in Dolan and Moré's paper [3]. Our profiles are based on the metrics f_{avg} , f_{best} and $CPUtime_{avg}$, the average of the CPU times required over all the 30 runs. We now briefly describe the main ideas behind the comparison based on profiles. Let \mathcal{P} and \mathcal{S} be the set of problems and the set of solvers in comparison, respectively, and $m_{p,s}$ be the performance metric used when solving problem $p \in \mathcal{P}$ by solver $s \in \mathcal{S}$. The relative comparison is based on the performance ratios defined by

$$r_{p,s} = \begin{cases} 1 + \frac{m_{p,s} - \min\{m_{p,s} : s \in \mathcal{S}\}}{\min\{m_{p,s} : s \in \mathcal{S}\}}, & \text{if } \min\{m_{p,s} : s \in \mathcal{S}\} < 0.00001 \\ \frac{m_{p,s}}{\min\{m_{p,s} : s \in \mathcal{S}\}}, & \text{otherwise} \end{cases} \quad (11)$$

and the overall assessment of the performance of a particular solver s is given by

$$\rho_s(\tau) = \frac{\text{no. of problems where } r_{p,s} \leq \tau}{\text{total no. of problems}}. \quad (12)$$

Thus, $\rho_s(\tau)$ gives the probability, for solver $s \in \mathcal{S}$, that $r_{p,s}$ is within a factor $\tau \in \mathbb{R}$ of the best possible ratio. The value of $\rho_s(1)$ gives the probability that the solver s will win over the others in the set. The solver which attains the least value of the performance metric mostly, has a higher $\rho_s(1)$. The higher the ρ_s the better the solver is. On the other hand, $\rho_s(\tau)$ for large values of τ measures the solver robustness.

4.2. Comparing the performance of fitness functions

Here we aim to compare the performance of the four fitness functions described in Subsection 3.3. using the competitive ranking method embedded in the AFS algorithm. Figure 1 contains two plots with the performance profiles obtained of the four functions: Φ_1 in Eq. (6), Φ_2 in Eq. (7), Φ_3 in Eq. (8) and Φ_4 in Eq. (9). From the plot on the left, based on the average performance, we conclude that the version with Φ_2 outperforms the other three in 59% of the tested problems. This means that in 59% of the problems the values of f_{avg} - the metric $m_{p,s}$ in these profiles - obtained by the function Φ_2 are better or equal to those obtained by the other fitness functions in comparison. Their performance ratios $r_{p,s}$ are then equal to one (see Eq. (11)). We remark that the metric f_{avg} is indeed the most important when comparing stochastic algorithms, since it reports the central tendency of the results over the runs. The metric f_{best} has also been used in the literature. However, the best of all solutions obtained over the runs is always biased since it is always smaller than all the remaining solutions [1].

Nevertheless, we include the plot on the right of Figure 1 to show the profiles based on f_{best} . Both fitness functions Φ_1 and Φ_2 give the best solutions for almost 42% of the tested problems.

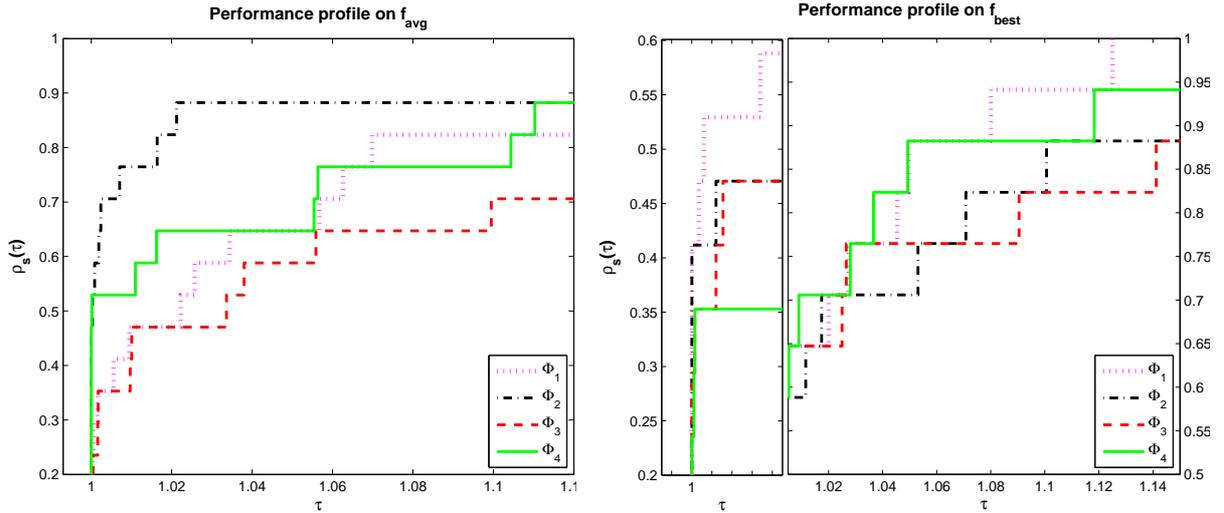


Figure 1: Performance profiles on f_{avg} and f_{best} .

Now, we aim to compare the computation effort of the four fitness functions using $CPUtime_{avg}$ (the average of the CPU times) as the performance metric. See Figure 2. Clearly Φ_1 is the least time consuming algorithm followed by the fitness Φ_3 and then Φ_2 . However, the average solutions accuracy attained by functions Φ_1 and Φ_3 are rather lower than that attained by Φ_2 . In less than 20% of the tested problems, the functions Φ_1 and Φ_3 are able to reach a value of f_{avg} that is equal or smaller than the other fitness functions in comparison. See the plot on the left of Figure 1.

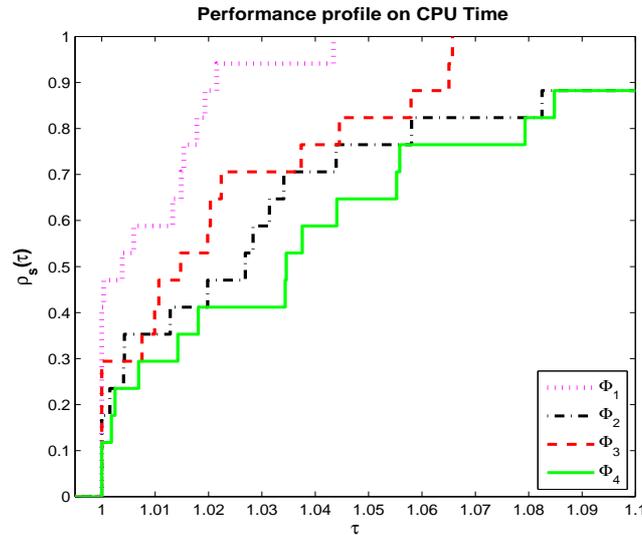


Figure 2: Comparison based on CPU time

5. Conclusions

This paper presents a competitive ranking method, which is used to rank the points in a population with respect to objective function and constraint violation separately, and is embedded in the Artificial Fish Swarm algorithm for solving constrained global optimization problems. A stochastic fitness function to assess the relative goodness of the points in the population is proposed and compared with other proposals found in the literature.

Computational tests carried out with a set of well-known global optimization problems show that the

proposed global competitive scheme to handle the equality and inequality constraints when embedding in the AFS algorithm is able to effectively solve constrained problems.

References

- [1] M. Birattari, and M. Dorigo, How to assess and report the performance of a stochastic algorithm on a benchmark problem: mean or best result on a number of runs?, *Optimization Letters*, 1, 309–311, 2007.
- [2] K. Deb, An efficient constraint-handling method for genetic algorithms, *Computer Methods in Applied Mechanics and Engineering*, 186 (2-4): 311–338, 2000.
- [3] E.D. Dolan, and J.J. Moré, Benchmarking optimization software with performance profiles, *Mathematical Programming*, 91, 201–213, 2002..
- [4] E.M.G.P. Fernandes, T.F.M.C. Martins and A.M.A.C. Rocha, Fish swarm intelligent algorithm for bound constrained global optimization, *Proc. of the 2009 CMMSE*, J.V. Aguiar (ed.), ISBN: 978-84-612-9727-6, 461–472, 2009.
- [5] A. Fiacco and G. McCormick, The sequential unconstrained minimization technique for nonlinear programming a primal- dual method, *Management Science*, 10, 360-366, 1964.
- [6] A. Homaifar, C. X. Qi, and S. H. Lai. Constrained optimization via genetic algorithms. *Simulation*, 62 (4), 242-253, 1994.
- [7] M. Jiang, Y. Wang, S. Pfletschinger, M.A. Lagunas and D. Yuan, Optimal multiuser detection with artificial fish swarm algorithm, *CCIS 2, ICIC 2007*, D.-S. Huang, L. Heutte and M. Loog (eds.), Springer-Verlag, 1084–1093, 2007.
- [8] M. Jiang, N. Mastorakis, D. Yuan and M.A. Lagunas, Image segmentation with improved artificial fish swarm algorithm, *Lecture Notes in Electrical Engineering*, 28, Proceedings of ECC, N. Mastorakis, V. Mladenov, V.T. Kontargyri (Eds.), ISBN: 978-0-387-84818-1, Springer-Verlag, 133–138, 2009.
- [9] J. A. Joines and C. R. Houck. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs. In David Fogel, editor, *Proceedings of the first IEEE Conference on Evolutionary Computation (CEC)*, volume 2, 579-584, Orlando, Florida, 1994.
- [10] Z. Michalewicz and N. F. Attia. Evolutionary optimization of constrained problems. *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, 98-108. World Scientific, 1994.
- [11] P.Y. Ho and K. Shimizu, Evolutionary constrained optimization using an addition of ranking method and a percentage-based tolerance value adjustment scheme. *Information Sciences*, 177 (14), 2985-3004, 2007.
- [12] F. Hoffmeister and J. Sprave. Problem-independent handling of constraints by use of metric penalty functions. In Lawrence J. Fogel, Peter J. Angeline, and Thomas Back, editors, *Proceedings of the Fifth Annual Conference on Evolutionary Programming (EP96)*, 289-294, San Diego, California, February 1996.
- [13] J.J. Liang, T.P. Runarsson, E. Mezura-Montes, M. Clerc, P.N. Suganthan, C.A.C. Coello, and K. Deb, *Problem definitions and evaluation criteria for the CEC2006 special session on constrained real-parameter optimization*, 2006.
(http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC-06/CEC06.htm)
- [14] A.M.A.C. Rocha, T.F.M.C. Martins and E.M.G.P. Fernandes, An augmented Lagrangian fish swarm based method for global optimization, *Journal of Computational and Applied Mathematics*, 2010. doi:10.1016/j.cam.2010.04.020
- [15] T.P. Runarsson and X. Yao, Stochastic Ranking for Constrained Evolutionary Optimization, *IEEE Transactions on Evolutionary Computation*, 4 (3), 274–283, 2000.

- [16] T.P. Runarsson and X. Yao, Chapter 4. Constrained Evolutionary Optimization: The penalty function approach. *Evolutionary Optimization*, (Editors R. Sarker, M. Mohammadian and X. Yao) Kluwer Academic Publishers, USA, 87-113, 2002. (ISBN: 0-7923-7654-4)
- [17] P. D. Surry and N. J. Radcliffe. The COMOGA method: Constrained optimisation by multi-objective genetic algorithms. *Control and Cybernetics*, 26 (3), 391-412, 1997.
- [18] C.-R. Wang, C.-L. Zhou, J.-W. Ma, An improved artificial fish-swarm algorithm and its application in feed-forward neural networks, *Proceedings of the 4th ICMLC*, 2890–2894, 2005.
- [19] Y. Wang, Z. Cai, Y. Zhou, and Z. Fan. Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique, *Structural and Multidisciplinary Optimization*, 37 (4), 395-413, 2009.
- [20] X. Wang, N. Gao, S. Cai, M. Huang, An artificial fish swarm algorithm based and ABC supported QoS unicast routing scheme in NGI, *Lecture Notes in Computer Science*, 4331, ISPA 2006 G. Min et al.(eds.), Springer-Verlag, 205–214, 2006.