Distribution based artificial fish swarm in continuous global optimization

Ana Maria A.C. Rocha[*], M. Fernanda P. Costa[†], Edite M.G.P. Fernandes[*]

[*] *Algoritmi R&D Centre, University of Minho, Portugal*
E-mail: {arocha;emgpf}@dps.uminho.pt

[†] *Centre of Mathematics, University of Minho, Portugal*
E-mail: mfc@math.uminho.pt

Abstract

Distribution based artificial fish swarm (DbAFS) is a new heuristic for continuous global optimization. Based on the artificial fish swarm paradigm, the new algorithm generates trial points from the Gaussian distribution, where the mean is the midpoint between the current and the target point and the standard deviation is the difference between those two points. A local search procedure is incorporated into the algorithm aiming to improve the quality of the solutions. The performance of the proposed DbAFS is investigated using a set of small bound constrained optimization problems.

Key words: Global optimization, artificial fish swarm, Gaussian distribution, local search.

1 Introduction

The artificial fish swarm (AFS) algorithm belongs to the class of stochastic population-based methods for solving continuous global optimization problems. The artificial fish is a fictitious entity of a true fish. Its movements are simulations and interpretations of fish behavior [Wang et al., 2006, Jiang et al., 2007, Xiu-xi et al., 2010, Rocha et al., 2011, Rocha and Fernandes, 2011, Neshat et al., 2013]. The environment in which the artificial fish moves, searching for the optimal solution of an optimization problem, is the feasible search space of the problem. In nature, fishes desire to stay close to the swarm, protecting themselves from predators and looking for food, and to avoid collisions within the group. These behaviors inspire mathematical modelers that need to solve efficiently optimization problems.

The main fish swarm behavior are the following:

- *random* behavior - in general, fish swims randomly in water looking for food and other companions;
- *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 goes directly and quickly to that region;
- *swarming* behavior - when swimming, fish naturally assembles in groups which is a living habit in order to guarantee the existence of the swarm and avoid dangers;
- *chasing* behavior - when a fish, or a group of fishes, in the swarm discovers food, the others in the neighborhood find the food dangling quickly after it.

The problem to be addressed in this paper is the bound constrained problem:

$$\min_{x\in\Omega} f(x), \tag{1.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a nonlinear function and $\Omega = \{x \in \mathbb{R}^n : -\infty < l_i \leq x_i \leq u_i < \infty,\ i = 1, \ldots, n\}$ 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. Here, our purpose is to compute a global minimizer, i.e., a point $x^* \in \Omega$ such that $f(x^*) \leq f(x)$ for all $x \in \Omega$. Many derivative-free algorithms and heuristics have been proposed to solve problem (1.1), namely those based on swarm intelligence [Engelbrecht, 2005]. Probably the most well-known and widely used in applications are

- the particle swarm optimization (PSO) algorithms [Kennedy and Eberhart, 1995, Ali and Kaelo, 2008, Coelho et al., 2005, Hao and Hu, 2009, Miranda et al., 2007, Pires et al., 2010, Vaz and Vicente, 2007];
- the bee colony [Bernardino et al., 2013, Karaboga and Basturk, 2007, Karaboga and Akay, 2009a), Karaboga and Akay, 2009b), Diwold et al., 2011]; and
- the ant colony algorithms [Dorigo and Stützle, 2004, Matos and Oliveira, 2004, Melo et al., 2010, Monteiro et al., 2013, Socha and Dorigo, 2008, Vilarinho and Simaria, 2006, Xiao and Li, 2011].

Our proposal for globally solving the problem (1.1) is a distribution based AFS, hereafter denoted by DbAFS, that takes the AFS algorithm presented in [Rocha et al., 2011] as a heuristic basis. The idea presented in this DbAFS algorithm is borrowed from the bare bones swarm concept [Kennedy, 2003, Omran et al., 2008]. The novelty in DbAFS algorithm is that each trial point is sampled from a Gaussian distribution, instead of being randomly generated in basis of the corresponding current point and a direction of search. The center of the distribution is given by the midpoint between the current and a target point, featured by the selected fish behavior, and the dispersion is related with the difference between those two points. The goal here is to improve the accuracy and convergence behavior.

The remainder of the paper is organized as follows. Section 2 describes the proposed DbAFS algorithm and Section 3 presents and discusses the results of preliminary experiments. Finally, the conclusions of this study and the ideas for future work are presented in Section 4.

2 Distribution based artificial fish swarm

The AFS algorithm is a stochastic method that relies on a swarm intelligence based paradigm to construct fish/point movements over the search space [Jiang et al., 2007, Neshat et al., 2013]. We will use the words ‘fish’ and ‘point’ interchangeably throughout the paper. Each point in the space is represented by $x^j \in \mathbb{R}^n$ (the jth point of a population), m is the number of points in the population, where $m < \infty$, and the component i of a vector x^j is represented by x^j_i.

2.1 The AFS algorithm

We now describe the procedure used to generate trial points in the AFS algorithm [Rocha et al., 2011]. At each iteration, a population of m solutions/points, herein denoted by $x^1, x^2, \ldots, x^m$ is used to generate a set of trial points $y^1, y^2, \ldots, y^m$. The population is initialized randomly in the entire search space Ω using the equation: $x^j_i = l_i + U(0,1)(u_i - l_i)$ for each component $i = 1, \ldots, n$ of the point x^j, where the notation $U(0,1)$ represents a random number in $(0,1)$.

Each point x^j movement is defined according to the number of points inside its ‘visual scope’. The ‘visual scope’ is defined as the closed neighborhood centered at x^j with a positive radius α. In the herein implemented versions of the AFS algorithm, the radius is dynamically defined as a fraction of the maximum distance between x^j and the other points x^l, $l \neq j$, $\alpha^j = \gamma \max_{l\neq j} \left\|x^j - x^l\right\|$, for $\gamma \in (0,1)$. Three possible situations may occur:

- the ‘visual scope’ is empty;
- the ‘visual scope’ is crowded;
- the ‘visual scope’ is not crowded.

When the ‘visual scope’ is empty, a *random* behavior is performed, in which the trial y^j is randomly generated inside the ‘visual scope’ of x^j.

When the ‘visual scope’ is crowded, with more than 80% of the population inside the ‘visual scope’ of x^j, a target point is randomly selected from the visual, x_{rand}. Then, if $f(x_{\text{rand}}) \leq f(x^j)$, the *searching* behavior is implemented and the trial point y^j is randomly defined along the direction from x^j to x_{rand}. Otherwise, the *random* behavior is performed.

When the ‘visual scope’ is not crowded, and $f(x_{\min}) \leq f(x^j)$, where $x_{\min}$ is the best point inside the ‘visual scope’, the *chasing* behavior is performed. This means that y^j is randomly generated along the direction from x^j to the target point $x_{\min}$. However, if $f(x_{\min}) > f(x^j)$ and the central point of the ‘visual scope’, denoted by $\bar{x}$, satisfies $f(\bar{x}) \leq f(x^j)$, then the *swarming* behavior is implemented instead. This means that the trial point is randomly defined along the direction from x_j to the target point $\bar{x}$. On the other hand, if $f(\bar{x}) > f(x^j)$, a target x_{rand} is randomly selected from the ‘visual scope’ and if $f(x_{\text{rand}}) \leq f(x^j)$, the *searching* behavior is carried out as previously described; otherwise the *random* behavior is performed.

We note that each point x^j generates a trial point y^j, inside the set Ω, as follows:

$$y^j_i = \max\left\{l_i, \min\left\{x^j_i + U(0,1)\, d^j_i, u_i\right\}\right\}, \text{ for } i = 1, \ldots, n \tag{2.1}$$

where d^j is a vector defined by the direction from x^j to one of the above referred target points.

To choose which point between the current x^j and the trial y^j will be a point of the population for the next iteration, the objective function at the two points are compared and if $f(y^j) \leq f(x^j)$ the trial point is passed to the next iteration as a current point. Otherwise, the current point is preserved to the next iteration/population.

2.2 Local search

In general, algorithms for globally solving an optimization problem have two separate phases:

- the global phase performs the exhaustive exploration of the search space using mostly stochastic methods to search for a promising region where a global minimum exists;
- the local phase exploits locally the promising region, using preferentially a deterministic method.

In the described AFS algorithm, a derivative-free deterministic method that exploits the neighborhood of a point for a better approximation using an exploratory move as well as a pattern move, and known as the Hooke and Jeeves (HJ) method [Hooke and Jeeves, 1961], is implemented.

This is a variant of the well-known coordinate search method. It incorporates a pattern move to accelerate the progress of the algorithm, by exploiting information obtained from the search in previous successful iterations. At the initial iteration, the exploratory move carries out a coordinate search centered at the best point of the population, with a step size of δ_{ls}. If a new trial point, y, with a better function value than x^{best} is encountered, the iteration is successful and δ_{ls} is maintained. Otherwise, the iteration is unsuccessful and δ_{ls} is reduced. If the previous iteration was successful, the vector $y - x^{\text{best}}$ defines a promising direction and a pattern move is then implemented, i.e., the exploratory move is carried out centered at the trial point $y+(y-x^{\text{best}})$, rather than at 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 centered at y. We refer to [Hooke and Jeeves, 1961] for details.

When the step size falls below δ_{tol}, a small positive tolerance, the search terminates since convergence has been attained [Kolda et al., 2003].

The pseudo-code for the AFS algorithm is presented below in Algorithm 1.

Algorithm 1 (AFS algorithm)

```
Data: $m \geq 1$, $0 < \gamma < 1$, $0 < \delta_{tol} \ll 1$.
1: Randomly generate $m$ points in $\Omega$, evaluate $x^j$, $j = 1, \ldots, m$, and select $x^{\text{best}}$.
2: While stopping conditions are not satisfied do
3:    For all $x^j$, $j = 1, \ldots, m$ do
4:       Generate trial point $y^j$ using (2.1), evaluate $y^j$, $j = 1, \ldots, m$.
5:       If $f(y^j) \leq f(x^j)$ then set $x^j = y^j$.
6:    End for
7:    Select $x^{\text{best}}$.
8:    Perform local search on $x^{\text{best}}$.
9: End while
```

2.3 Distribution based artificial fish swarm

The distribution based artificial fish swarm is evolved from a classical version of the AFS algorithm [Rocha et al., 2011]. A Gaussian sampling strategy is proposed to create the trial points. After selecting the fish behavior to be implemented to each current point of the population, x^j, each component i ($i = 1, \ldots, n$) of the corresponding trial point, y_i^j, is randomly selected from a Gaussian distribution $N(\mu_i, \sigma_i)$, where:

- the mean is given by the average of the corresponding components of the current and the target point, $\mu_i = \dfrac{x_i^j + t_i}{2}$;
- the standard deviation is the absolute difference between the corresponding components of the current and the target point, $\sigma_i = \left| x_i^j - t_i \right|$;

and the target point t is x_{rand} if the *searching* behavior is selected, $x_{\min}$ if *chasing* is selected, or $\bar{x}$ if *swarming* is selected. Furthermore, to maintain the trial point inside the set Ω, the following equation is used:

$$y_i^j = \max\left\{ l_i, \min\left\{ y_i^j, u_i \right\} \right\}, \text{ for } i = 1, \ldots, n. \tag{2.2}$$

However, when a point x^j follows a *random* behavior, it will have a 50% chance that the component i of the point changes to the corresponding component of the best point of the population:

$$y_i^j = \begin{cases} x_i^j & \text{if } U(0,1) > 0.5 \\ x_i^{\text{best}} & \text{otherwise} \end{cases} . \tag{2.3}$$

We note that the exploration feature of the algorithm is ensured by the standard deviation which converges to zero as the points in the population converge to the optimal solution of the problem.

3 Numerical experiments

For a preliminary practical assessment of the proposed DbAFS algorithm, numerical experiments are carried out involving nine standard benchmark small test problems with known acronyms [Ali et al., 2005]: BR (n=2), CB6 (n=2), GP (n=2), H3 (n=3), H6 (n=6), SBT (n=2), S5, S7 and S10 (n=4). The algorithm is coded in C and the results are obtained on a PC with a 2.8 GHz Core Duo Processor P9700 and 6 Gb of memory. Each problem was solved 30 times and a population of $m = 10n$ points is used. After an empirical study, we also set $\gamma = 0.8$ and $\delta_{tol} = 10^{-8}$. The initial δ_{ls} is set to $10^{-3} \max_i\{u_i - l_i\}$.

The performance of the DbAFS algorithm is compared with the version 'AFS' algorithm described in [Rocha et al., 2011], the improved particle swarm optimization (PSO) algorithms - variants 'PSO-RPB' and 'PSO-HS' - proposed in [Ali and Kaelo, 2008], the differential evolution 'DE' (originally presented in [Storn and Price, 1997]), and 'MADDF' described in [Fernandes et al., 2012]. A brief description of the notation used in this section follows:

- 'DbAFS$_{HJ}$' is the distribution based AFS algorithm with the HJ local procedure (Section 2.2);
- 'DbAFS$_{rand}$' is the distribution based AFS algorithm with a simple random local search procedure;
- 'DbAFS' is the distribution based AFS algorithm without the local search;
- 'AFS$_{HJ}$' corresponds to the version presented in [Rocha et al., 2011], with the HJ local procedure;
- 'PSO-RPB' is a version of PSO with randomized *pbest* (best position of a particle) in the cognitive component;
- 'PSO-HS' is a variant of the PSO algorithm with a hybrid scheme (using the point generation scheme of DE) for position update;
- 'DE' is an improved version of the original differential evolution where the scaling factor is randomly chosen, for each point, and the crossover parameter is randomly chosen, for each iteration;
- 'MADDF' is a derivative-free multistart technique based on a filter set methodology for constraint-handling. (Bound constraints are incorporated into a constraint violation function.)

The stopping conditions used to stop the algorithms are [Ali and Kaelo, 2008]:

$$\left|f(x^{\text{best}}) - f_{\text{opt}}\right| \leq 0.001 \quad \text{or} \quad nf_{\text{eval}} > 20000, \tag{3.1}$$

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. However, if the optimal solution of the problem is unknown, the algorithm may use other termination conditions.

First, we aim to analyze the effect on the performance of the DbAFS algorithm when a local search procedure is included in the algorithm (see Step 8 in Algorithm 1). Figure 1 depicts the average nf_{eval} after the 30 runs, required by DbAFS and DbAFS$_{HJ}$ for each problem. We observe that the HJ local search has improved the efficiency of the algorithm when solving eight out of the nine tested problems. Only for the problem H6, DbAFS$_{HJ}$ required in average more function evaluations than DbAFS to reach the solution with the accuracy specified in (3.1).

With an illustrative purpose, Figure 2 aims to compare the DbAFS$_{HJ}$ algorithm performance for different parameter values. The plot on the left shows the convergence behavior for three values of the factor γ in the definition of the 'visual scope', on the problem H6: $\gamma = \{0.1, 0.5, 0.8\}$. The largest value of γ gives the fastest reduction in f when applied to H6. The plot on the right aims to show how population size affects the convergence of the algorithm. The tested values are $m = \{10, 10n, 100\}$ and the experiment was done on the problem H6 with $\gamma = 0.8$. Convergence is rather slow with the largest value of the population size.

Finally, Table 1 summarizes the results obtained in terms of the average number of function evaluations required by the algorithms to reach the optimal solution with the accuracy defined in (3.1). (The reported results of 'PSO-RPB', 'PSO-HS', 'DE' and 'MADDF' are adopted directly from [Ali and Kaelo, 2008, Fernandes et al., 2012].) Based on the results we may conclude that DbAFS$_{HJ}$ has a good performance. The computational effort in terms of function evaluations has been reduced when compared with the

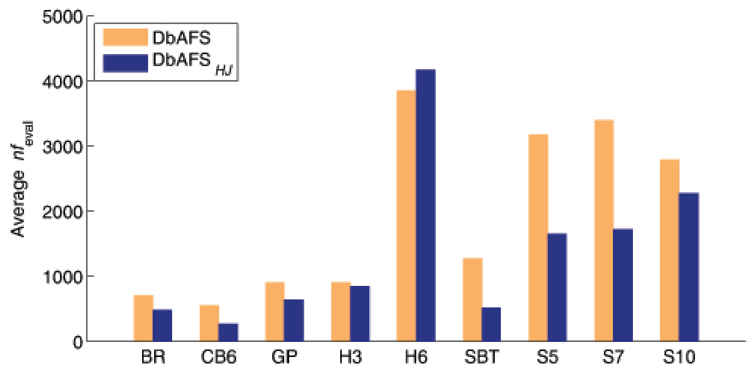

Figure 1: Average nf_{eval} for the nine problems: DbAFS vs. DbAFS$_{HJ}$.

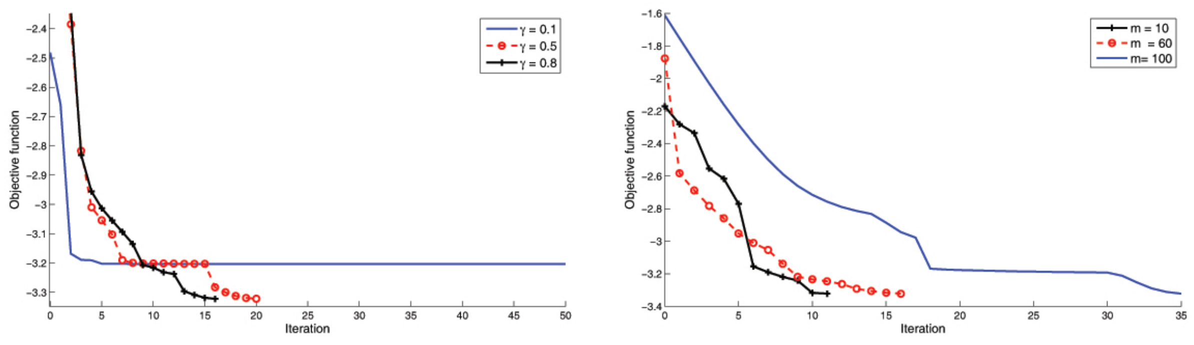

Figure 2: Factor γ effect (left) and population size effect (right), on the problem H6.

Table 1: Results of average number of function evaluations.

Prob.	f_{opt}	DbAFS		other methods				
		DbAFS$_{HJ}$	DbAFS$_{rand}$	AFS$_{HJ}$	PSO-RPB	PSO-HS	DE	MADDF
BR	0.39789	487	690	651	2652	2018	1305	506
CB6	-1.0316	274	293	246	2561	2390	1127	660
GP	3.00000	642	710	562	2817	1698	884	1063
H3	-3.86278	851	911	1573	3564	2948	1238	5845
H6	-3.32237	4167	3864	7861	8420	8675	7053	7559
SBT	-186.731	526	1256	659	4206	6216	2430	1867
S5	-10.1532	1650	1611	3773	6641	6030	5824	2929
S7	-10.4029	1723	1818	2761	6860	6078	5346	4428
S10	-10.5364	2282	1889	2721	6747	5602	4822	4489

basic AFS$_{HJ}$. The HJ local procedure incorporated into the DbAFS algorithm has improved in average the convergence to the optimal solution in six out of the nine tested problems, when compared with a simple random local search. The results also show that the DbAFS$_{HJ}$ performs better than the other methods in comparison (PSO variants, DE and MADDF).

4 Conclusions

This paper has introduced a new AFS algorithm which uses the Gaussian distribution to create the trial points, at every iteration. The main features of AFS are maintained, in particular the concept of 'visual scope' of each point and the selection of fish behavior. The proposed DbAFS algorithm performs well in a set of small benchmark problems, in terms of number of function evaluations. The effect of some parameters on the performance of the algorithm has been analyzed. A comparison with other stochastic population-based techniques available in the literature is carried out using small dimensional problems.

The results are promising in terms of reduced computational effort.

There are still some issues related to parameter setting that need further investigation. Extensive numerical experiments with larger dimensional problems remain to be done and will be reported in a future paper.

Acknowledgments

This work was financed by FEDER funds through COMPETE (Operational Programme Thematic Factors of Competitiveness) and by portuguese funds through FCT (Foundation for Science and Technology) within the projects FCOMP-01-0124-FEDER-022674 and PEst-C/MAT/UI0013/2011.

References

[Ali and Kaelo, 2008] Ali, M.M and Kaelo, P. (2008). Improved particle swarm algorithms for global optimization. *Applied Mathematics and Computation*, 196:578–593.

[Ali et al., 2005] Ali, M.M., Khompatraporn, C., and Zabinsky, Z. B. (2005). A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. *Journal of Global Optimization*, 31:635–672.

[Bernardino et al., 2013] Bernardino, E.M., Bernardino, A.M., Sánchez-Pérez, J.M., Gómez-Pulido, J.A., and Vega-Rodríguez, M.A. (2013). Swarm optimization algorithms applied to large balanced communication networks. *Journal of Network and Computer Applications* 36(1) 504–522.

[Coelho et al., 2005] Coelho, J.P., Oliveira, P.B.M., and Cunha, J.B. (2005). Greenhouse air temperature predictive control using the particle swarm optimisation algorithm. *Computers and Electronics in Agriculture*, 49(3):330–344.

[Diwold et al., 2011] Diwold, K., Aderhold, A., Scheidler, A., and Middendorf, M. (2011). Performance evaluation of artificial bee colony optimization and new selection schemes. *Memetic Computing*, 3:149–162.

[Dorigo and Stützle, 2004] Dorigo, M. and Stützle, T. (2004). Ant Colony Optimization. MIT Press, Cambridge, Mass, USA.

[Engelbrecht, 2005] Engelbrecht, A. (2005). Fundamentals of Computational Swarm Intelligence. Wiley & Sons, New York.

[Fernandes et al., 2012] Fernandes, F.P., Costa, M.F.P., and Fernandes, E.M.G.P. (2012). A derivative-free filter driven multistart technique for global optimization. *Lecture Notes in Computer Science*, 7335, ICCSA 2012 Part III, Murgante, B. et al.(Eds.) 103–118.

[Hao and Hu, 2009] Hao, L. and Hu, L. (2009). Hybrid particle swarm optimization for continuous problems. *ISECS International Colloquium on CCCM 2009*, 3:217–220.

[Hooke and Jeeves, 1961] Hooke, R. and Jeeves, T.A. (1961). Direct search solution of numerical and statistical problems. *Journal of the Association for Computing Machinery*, 8:212–229.

[Jiang et al., 2007] Jiang, M., Wang, Y., Pfletschinger, S., Lagunas, M.A., and Yuan, D. (2007). Optimal multiuser detection with artificial fish swarm algorithm. In *CCIS Vol. 2*, ICIC 2007 Part 22, Huang, D.-S. et al.(Eds.) 1084–1093.

[Karaboga and Akay, 2009a)] Karaboga, D. and Akay, B. (2009). A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation*, 214(1):108–132.

[Karaboga and Akay, 2009b)] Karaboga, D. and Akay, B. (2009). A survey: algorithms simulating bee swarm intelligence. *Artificial Intelligence Review*, 31:61–85.

[Karaboga and Basturk, 2007] Karaboga, D. and Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, 39(3):459–471.

[Kennedy, 2003] Kennedy, J. (2003). Bare bones particle swarms. In *Proc. IEEE Swarm Intelligence Symposium* 80–87.

[Kennedy and Eberhart, 1995] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proc. IEEE Int. Conf. Neural Network 1995* 1942–1948.

[Kolda et al., 2003] Kolda, T.G., Lewis, R.M., Torczon, V. (2003). Optimization by Direct Search: New Perspectives on Some Classical and Moddern Methods. *SIAM Review*, 45(3):385–482.

[Matos and Oliveira, 2004] Matos, A.C. and Oliveira, R.C. (2004). An experimental study of the ant colony system for the period vehicle routing problem. In *Lecture Notes in Computer Science* 3172, ANTS 2004, M. Dorigo et al. (Eds.) 286–293.

[Melo et al., 2010] Melo, L., Pereira, F., and Costa, E. (2010). MC-ANT: A multi-colony ant algorithm. In *Lecture Notes in Computer Science* 5975, EA 2009, P. Collet et al. (Eds.) 25–36.

[Miranda et al., 2007] Miranda, V., Keko, H., and Jaramillo, A. (2007). EPSO: evolutionary particle swarms. *Studies in Computational Intelligence*, 66:139–167.

[Monteiro et al., 2013] Monteiro, M.S.R., Fontes, D.B.M.M., and Fontes, F.A.C.C. (2012). Concave minimum cost network flow problems solved with a colony of ants. *Journal of Heuristics*, 19(1):1–33.

[Neshat et al., 2013] Neshat, M., Sepidnam, G., Sargolzaei, M., and Toosi, A.N. (2013). Artificial fish swarm algorithm: a survey of the state-of-the-art, hybridization, combinatorial and indicative applications. *Artificial Intelligence Review*, DOI:10.1007/s10462-012-9342-2 (2013).

[Omran et al., 2008] Omran, M.G.H., Engelbrecht, A.P., and Salman, A. (2008). Bare bones differential evolution. *European Journal of Operational Research*, 196(1):128–139.

[Pires et al., 2010] Pires, E.J.S., Machado, J.A.T., Oliveira, P.B.M., Cunha, J.B., and Mendes, L. (2010). Particle swarm optimization with fractional-order velocity. *Nonlinear Dynamics*, 61(1-2):295–301.

[Rocha et al., 2011] Rocha, A.M.A.C., Fernandes, E.M.G.P., and Martins, T.F.M.C. (2011). Novel fish swarm heuristics for bound constrained global optimization problems. *Lecture Notes in Computer Science*, 6784, ICCSA 2011 Part III B. Murgante, B. et al.(Eds.) 185–199.

[Rocha and Fernandes, 2011] Rocha, A.M.A.C. and Fernandes, E.M.G.P. (2011). Mutation-Based Artificial Fish Swarm Algorithm for Bound Constrained Global Optimization. In *Numerical Analysis and Applied Mathematics ICNAAM 2011*: AIP Conf. Proc. 1389, 751–754.

[Socha and Dorigo, 2008] Socha, K. and Dorigo, M. (2008). Ant colony optimization for continuous domains. *European Journal of Operational Research*, 185(3):1155–1173.

[Storn and Price, 1997] Storn, R. and Price, K. (1997). Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359.

[Vaz and Vicente, 2007] Vaz, A.I.F. and Vicente, L.N. (2007). A particle swarm pattern search method for bound constrained global optimization. *Journal of Global Optimization*, 39:197–219.

[Vilarinho and Simaria, 2006] Vilarinho, P.M. and Simaria, A.S. (2006). ANTBAL: an ant colony optimization algorithm for balancing mixed-model assembly lines with parallel workstation. *International Journal of Production Research*, 44(2):291–303.

[Wang et al., 2006] Wang, X., Gao, N., Cai, S., and Huang, M. (2006). An artificial fish swarm algorithm based and ABC supported QoS unicast routing scheme in NGI. *Lecture Notes in Computer Science*, 4331, ISPA 2006, Min, G. et al.(Eds.) 205–214.

[Xiao and Li, 2011] Xiao, J., and LiangPing Li, L. (2011). A hybrid ant colony optimization for continuous domains. *Expert Systems with Applications*, 38(9):11072–11077.

[Xiu-xi et al., 2010] Xiu-xi, W., Hai-wen, Z., and Yong-quan, Z. (2010). Hybrid artificial fish school algorithm for solving ill-conditioned linear systems of equations. In*IEEE international conference on intelligent computing and intelligent systems (ICIS)*, 390—394.