An Enhanced Fish School Search Algorithm

C. J. A. Bastos-Filho, D. O. Nascimento
University of Pernambuco
Recife, Brazil
carmelofilho@ieee.org

***Abstract*—Fish School Search (*FSS*) is swarm-based optimizer that excels on multimodal search problems, but presents some drawbacks, such as the necessity to proper define the step used in some operators and the need to evaluate the fitness function twice per fish per iteration. This paper presents a simpler and enhanced version of the *FSS*, that features three advantages over the original *FSS*: high exploitation capability, just one fitness evaluation per fish per iteration and easy implementation. We name this novel version as *FSS-II*. Our proposal was compared to the *FSS* and the two most used *PSO* variations in terms velocity of convergence and robustness in six benchmark functions. *FSS-II* outperformed the other approaches in most of cases.**

***Index Terms*—Fish School Search, Swarm Intelligence, Optimization, Multimodal search spaces.**

I. Introduction

Swarm Intelligence has been widely used to tackle high dimensional optimization problems. Many swarm-based approaches have been proposed in the two last decades, such as: Ant Colony Optimization (*ACO*) [1] [2], Particle Swarm Optimization (*PSO*) [3] [4] [5], Artificial Bee Colony (*ABC*) [6] [7], Firework Algorithm (*FWA*) [8] [9], Cuckoo Search [10], Herd Optimization [11], Firefly Optimization Algorithm [12], Fish School Search (*FSS*) [13] [14] [15] [16], among others.

Each one of these algorithms features interesting capabilities that can be applied to specific scenarios. As an example, the standard version of the *PSO* with global topology presents a fast convergence and good exploitation capability, but the swarm quickly looses diversity when a local optimum is found during the search process [5].

The FSS algorithm was introduced in 2008 aiming to tackle optimization tasks in multimodal search spaces [13]. The main advantage of this algorithm is the capability to self-regulate the trade-off between exploration and exploitation. Some preliminary results indicate that *FSS* can outperform some of the swarm-based algorithms in benchmark functions with multimodal search spaces [13]. Although *FSS* presents this interesting behavior for multimodal search, it presents three main drawbacks: low exploitation capability, necessity to evaluate the fitness function twice per iteration per fish and relative high complexity for implementation, specially when compared to other widely used Swarm Intelligence algorithms, such as *PSO*.

This paper presents a simpler and enhanced version of the *FSS*, that features three advantages over the original *FSS*: higher exploitation capability, just one fitness evaluation per fish per iteration and easier implementation. We name this novel version as *FSS-II*.

The remainder of the paper is organized as follows: in Section II we give an overview of the *FSS* algorithm. In Section III we present a novel version for the *FSS* algorithm, the *FSS-II*. In Section IV we detail the simulation setup, including the description of the deployed benchmark functions. In Section V, we present some simulation results comparing our proposal (*FSS-II*) to the original version of the *FSS* and the two most used versions of the *PSO*, in terms of velocity of convergence and robustness. In Section VI we discuss the results and give our conclusions.

II. The Original Fish School Search Algorithm

The *FSS* algorithm was inspired in the emergent behavior of fish schools searching food [13]. The search process in *FSS* is carried out by a population of simple reactive agents, the fish. The search is performed in a bounded search space, called aquarium. Each fish i is represented by a position within the search space, $\vec{x}_i(t)$, and its weight, $W_i(t)$. The weight of the fish is updated iteratively depending on the amount of food found in the aquarium. Food density is related to the fitness value within the search space. For minimization problems, the amount of food in a particular region is inversely proportional to the fitness value in this region. The original version of the *FSS* algorithm has four operators, which can be grouped in two classes: feeding and swimming. Feeding processes are used to update the weights of the fish, whereas the swimming operators drive the fish movements. The original *FSS* has three swimming operators: the individual movement, the collective instinctive movement and the collective volitive movement.

The individual movement is used to trigger the other operators. In this operator, each fish randomly chooses a new position in its neighborhood according to (1):

$$\vec{n}_i(t+1) = \vec{x}_i(t) + \vec{s}(t+1), \tag{1}$$

in which $\vec{s}(t+1)$ is random vector defined for each fish at each iteration in each dimension by $s_{ind}(t+1) \cdot rand[-1;1]$. $s_{ind}(t+1)$ is the individual step and it is predefined prior to the search process. In general, it decays linearly along the iterations. $rand[-1;1]$ is a random value generated by an uniform probability density function in the interval [-1;1].

After the individual movement, the feeding operator is executed. In the feeding operator, the neighbor position $\vec{n}_i(t+1)$ is evaluated in terms of the fitness function $f[\vec{n}_i(t+1)]$. For each fish, the difference between the fitness of the neighbor position $f[\vec{n}_i(t+1)]$ and the current position $f[\vec{x}_i(t)]$ is evaluated as shown in (2). This difference $\Delta f_i(t+1)$ is used to feed the fish i, *i.e.* update the weight $W_i(t+1)$, according to (3). This means that the weight of each fish increases depending on the success rate achieved by the individual movement. In the original version of the *FSS*, each fish performs a greedy search and moves to $\vec{n}_i(t+1)$ if the neighbor position is better than the current position.

$$\Delta f_i(t+1) = f[\vec{n}_i(t+1)] - f[\vec{x}_i(t)], \tag{2}$$

$$W_i(t+1) = W_i(t) + \frac{\Delta f_i(t+1)}{max[\Delta f(t+1)]}. \tag{3}$$

The first collective movement is called instinctive. This movement is a drift influenced only by fish that successfully performed individual movements. This drift, $\vec{m}(t+1)$, is evaluated by using (4):

$$\vec{m}(t+1) = \frac{\sum_{i=1}^{N} \Delta\vec{x}_i(t+1)\Delta f_i(t+1)}{\sum_{i=1}^{N} \Delta f_i(t+1)}, \tag{4}$$

in which $\Delta\vec{x}_i(t)$ is the displacement of the fish i generated by the individual movement.

This drift is then applied to update the positions of all fish according to (5):

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{m}(t+1). \tag{5}$$

The second collective movement is the volitive operator. This movement is based on the overall success rate of the entire fish school, *i.e.* the fish school weight, $W(t+1)$, which corresponds to the summation of all fish weights at iteration $t+1$. If $W(t+1)$ is increasing, *i.e.* $W(t+1) > W(t)$, this means that the search has been successful and the radius of the school should contract to increase the exploitation ability. On the other hand, if $W(t+1)$ is decreasing, the fish school should expand to allow more exploration of the search space. This movement is executed regarding the school barycenter, which is evaluated by (6).

$$\vec{B}(t+1) = \frac{\sum_{i=1}^{N} \vec{x}_i(t+1)W_i(t+1)}{\sum_{i=1}^{N} W_i(t+1)}. \tag{6}$$

The volitive movement is inward regarding $\vec{B}(t+1)$, if $W(t+1) > W(t)$. In this case, all fish are updated according to (7). If $W(t+1) < W(t)$, the fish positions are updated according to (8).

$$\vec{x}(t+1) = \vec{x}(t+1) - s_v(t+1) \cdot \frac{\vec{x}_i(t+1) - \vec{B}(t+1)}{dist[x_i(t+1), B(t+1)]}, \tag{7}$$

$$\vec{x}(t+1) = \vec{x}(t+1) + s_v(t+1) \cdot \frac{\vec{x}_i(t+1) - \vec{B}(t+1)}{dist[x_i(t+1), B(t+1)]}, \tag{8}$$

in which $\vec{s}_v(t+1)$ is random vector defined for each fish at each iteration in each dimension by $s_{vol}(t+1) \cdot rand[0;1]$. $s_{vol}(t+1)$ is the volitive step and it is twice the $s_{ind}(t+1)$ value in the original *FSS*. $dist[\cdot]$ is a function which returns the Euclidean distance.

Algorithm 1 depicts the pseudocode of the original *FSS* algorithm. One can observe that the fitness must be evaluated twice per fish per iteration, since each fish must evaluate the fitness at $\vec{x}_i(t)$ and $\vec{n}_i(t+1)$.

Algorithm 1 FSS Pseudocode.

```
1:  Initialize randomly all fish positions $\vec{x}_i(0)$;
2:  Initialize randomly all fish weights $\vec{w}_i(0)$;
3:  while stop criterion is not met do
4:    for each fish do
5:      Find neighbor position (1);
6:      Evaluate fitness according to (2) and perform greedy
        search;
7:      Feed the fish using (3);
8:    end for
9:    Evaluate the drift using (4);
10:   for each fish do
11:     Execute instinctive movement using (5);
12:   end for
13:   Calculate barycenter using (6);
14:   for each fish do
15:     Execute volitive movement using either (7) or (8);
16:   end for
17:   Update $s_{ind}$ and $s_{vol}$.
18: end while
```

III. Our Proposal: A Simpler and Adaptive Fish School Search Algorithm

As mentioned before, the *FSS* presents an interesting emergent behavior, which allows the swarm to self-adapt the exploration-exploitation balance during the search process. However, the *FSS* performance depends on the steps used in the individual and the volitive collective movements. If higher values are used for these steps, the *FSS* presents a low exploitation capability since the minimum granularity of the search is limited. On the other hand, if lower values are used for these steps, the convergence of the algorithm is too slow. Furthermore, each fish needs to evaluate the fitness function twice per iteration to drive the feeding process. Besides, the *FSS* presents three loops inside the main loop of the algorithm (see Algorithm 1). This kind of structure is more complex for implementation when compared to other simpler swarm-based algorithms, such as *PSO*.

We propose in this paper a novel version of the *FSS* algorithm, named *FSS-II*, which aims to mitigate these drawbacks presented by the original *FSS*. In our proposal, there is no need to define the steps used in the individual and collective volitive movements. In the *FSS-II* algorithm, the operators are combined in a single equation. Besides, the collective movements are triggered by information acquired in preceding iterations and the fitness function is just evaluated once per iteration per fish.

Algorithm 2 presents the pseudocode of our proposal. The first four command lines are only used to initialize the swarm and the optimization process. The fish are randomly initialized in a predefined region of the search space. Then, a randomly local search is applied to every fish to allow the evaluation of the first weight variation.

Algorithm 2 *FSS-II* Pseudocode.

```
1: Initialize randomly every fish positions $\vec{x}_i(0)$ and weights $\vec{W}_i(0)$;
2: Apply a local search to every fish to define $\vec{x}_i(1)$;
3: Evaluate the weights $W_i(1)$ based on $f[\vec{x}_i(1)] - f[\vec{x}_i(0)]$ and $\vec{W}_i(0)$;
4: $t = 1$;
5: while stop criterion is not met do
6:     for each fish do
7:         Evaluate fish displacement using (9);
8:         Evaluate fitness variation using (10);
9:         Feed the fish using (11);
10:        Evaluate weight variation using (12);
11:    end for
12:    Calculate barycenter using (13);
13:    for each fish do
14:        Update fish position using (14).
15:    end for
16: end while
```

In the main loop, the first command is used to evaluate the individual displacement, $\Delta\vec{x}_i(t+1)$, that corresponds to the individual movement in the original *FSS*. In this novel version, this displacement depends on the position variation between iterations $t-1$ and t. The displacement is evaluated according to (9).

$$\Delta\vec{x}_i(t+1) = [\vec{x}_i(t) - \vec{x}_i(t-1)]. \tag{9}$$

Then, the fitness variation is evaluated by using (10), which is used to feed the fish (11). After feeding, the weight variation for each fish is evaluated by using (12).

$$\Delta f_i(t+1) = f[\vec{x}_i(t)] - f[\vec{x}_i(t-1)], \tag{10}$$

$$W_i(t+1) = W_i(t) + \Delta f_i(t+1), \tag{11}$$

$$\Delta W_i(t+1) = W_i(t+1) - W_i(t). \tag{12}$$

We calculate the barycenter according to (13).

$$\vec{B}(t+1) = \frac{\sum_{i=1}^{N} \vec{x}_i(t) W_i(t+1)}{\sum_{i=1}^{N} W_i(t+1)}. \tag{13}$$

Then, the fish positions are updated according to (14). One can observe that there are four terms in equation (14). The first term correspond the current position. The second term is related to the individual movement and it is based on the displacement considering preceding iterations. The third term is related to the instinctive collective movement, but in this case, it is weighted by $W_i(t+1)$. The fourth term is related to the volitive collective movement, but not bounded by a predefined step.

$$\begin{aligned}\vec{x}_i(t+1) &= \vec{x}_i(t) + \beta \cdot c \cdot \Delta\vec{x}_i(t+1)\\ &+ c \cdot rand(0,1) \cdot \frac{\sum_{i=1}^{N} \Delta\vec{x}_i(t+1) W_i(t+1)}{\sum_{i=1}^{N} W_i(t+1)}\\ &+ c \cdot rand(0,1) \cdot sign[\sum_{i=1}^{N} \Delta W_i(t+1)] \cdot [\vec{x}_i(t) - \vec{B}(t+1)],\end{aligned} \tag{14}$$

in which $sign[\cdot]$ returns the signal of the argument. Observe that in this case the signal function defines the direction of a displacement regarding the barycenter. We use c to control the movement amplitude according to the succes of the search process. And β to control the fish individual contribution during the movement.

We used an adaptative mechanism to update c, guided by a rate α. We start updating c using the rule, $c = c \cdot (1-\alpha)$. In each iteration, we assess the number of fish that increased its weight. If this number decreases, we change the signal of α. For instance, if we are using $c = c \cdot (1-\alpha)$, we change the update rule to $c = c \cdot (1+\alpha)$. And if we are using $c = c \cdot (1+\alpha)$, we change the update rule to $c = c \cdot (1-\alpha)$. However, if the number of fishes increases, we maintain the current rule, keeping the signal of α.

IV. Simulations Setup

We performed our simulations in six well known benchmark minimization problems [5], [17]. The functions are the Sphere, Rosenbrock, Schwefel 1.2, Rastrigin, Griewank and Ackley, presented in Eq. (15), (16), (17), (18), (19) and (20), respectively.

$$F_{Sphere}(\vec{x}) = \sum_{i=1}^{n} x_i^2, \tag{15}$$

$$F_{Rosenbrock}(\vec{x}) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (1-x_i)^2\right], \tag{16}$$

$$F_{Shwefel1.2}(\vec{x}) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2, \tag{17}$$

$$F_{Rastrigin}(\vec{x}) = 10n + \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i)\right], \tag{18}$$

$$F_{Griewank}(\vec{x}) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right), \tag{19}$$

$$\begin{aligned}F_{Ackley}(\vec{x}) &= -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right)\\ &- \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e.\end{aligned} \tag{20}$$

The search space, the initialization subspace and optimum point in the search space for the used benchmark functions are shown in Table I. One must observe that we initialized the fish/particles far away from the optimum point in order to test the exploration capabilities of the algorithms.

TABLE I
THE SEARCH SPACE, THE INITIALIZATION SUBSPACE AND THE OPTIMUM POINT IN THE SEARCH SPACE FOR THE BENCHMARK FUNCTIONS USED IN THE SIMULATIONS.

Function	Search space	Initialization subspace	Optimum point
Sphere	$100 \leq x_i \leq 100$	$50 \leq x_i \leq 100$	0.0^D
Rosenbrock	$-30 \leq x_i \leq 30$	$15 \leq x_i \leq 30$	1.0^D
Schwefel 1.2	$-100 \leq x_i \leq 100$	$50 \leq x_i \leq 100$	0.0^D
Rastrigin	$-5.12 \leq x_i \leq 5.12$	$2.56 \leq x_i \leq 5.12$	0.0^D
Griewank	$-600 \leq x_i \leq 600$	$300 \leq x_i \leq 600$	0.0^D
Ackley	$-32 \leq x_i \leq 32$	$16 \leq x_i \leq 32$	0.0^D

In all simulations, we performed 30 trials per function. In each trial the stoping criterion was $300,000$ fitness evaluations in a 30 dimensions search space.

We used 30 fish for the *FSS* and for the *FSS-II*. In the original *FSS*, each fish has to evaluate the fitness twice per iteration. In both *FSS*-based approaches, we randomly initialized the weights of the fish in the interval $[300; 600]$. During the search process, we allowed the weights of the fish to vary in the interval $[1; 1,000]$. For the *FSS*, $s_{ind}(0) = 0.10$ of the search space, $S_{ind}(10,000) = 0.0001$ of the search space and $S_{vol} = 2 \cdot S_{ind}$. For the *FSS-II*, we performed some preliminary simulations and defined $\alpha = 0.01$ and $\beta = 0.4$. c was initialized with value equal to 0.1 and can vary in the interval $[0.001; 0.999]$.

Since Particle Swarm Optimization (*PSO*) is one of the most used swarm intelligence algorithms for continuous optimization [5], we used the two most used variations of the *PSO*, standard version with global topology (*GPSO*) and standard version with local topology (*LPSO*), for the sake of comparison. For *GPSO* and *LPSO*, we used 30 particles and $c_1 = c_2 = 2.05$. We used the inertia factor decaying linearly from 0.9 to 0.4 along the iterations.

V. SIMULATIONS RESULTS

Figures 1, 2 and 3 present the evolution of the average fitness value for the Sphere, Rosenbrock and Rastrigin functions, respectively. The Sphere function is used to assess the exploitation capability. From Figure 1, one can observe that the our proposal presents a high exploitation capability, even when compared to the *GPSO*. The Rosenbrock function is used to assess the capability of the algorithm to escape from a plateau. From Figure 2, one can observe that the our proposal got stuck in the first $5,000$ fitness evaluations, but escaped from the plateau and outperformed the other approaches from $20,000$ fitness evaluations. The Rastrigin function is a multimodal function in used to assess the capability of the algorithm to escape from local minima. From Figure 3, one can observe that the our proposal far outperformed the other approaches during the entire search process.

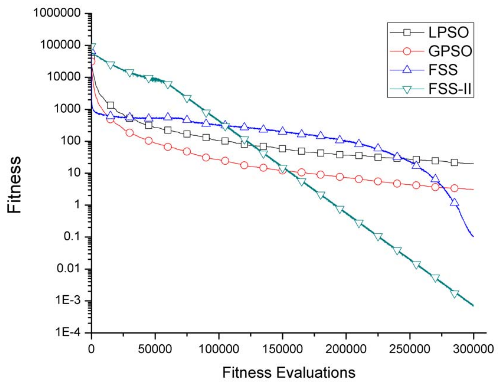

Fig. 1. Evolution of the average fitness value for the Sphere function.

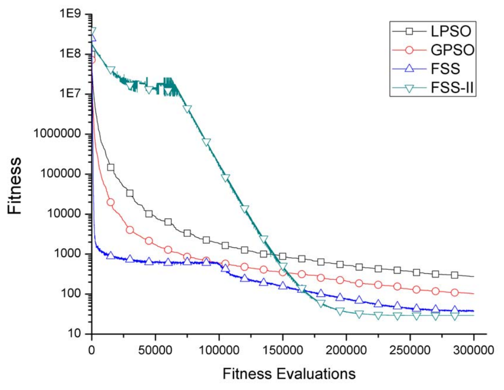

Fig. 2. Evolution of the fitness value for the Rosenbrock function.

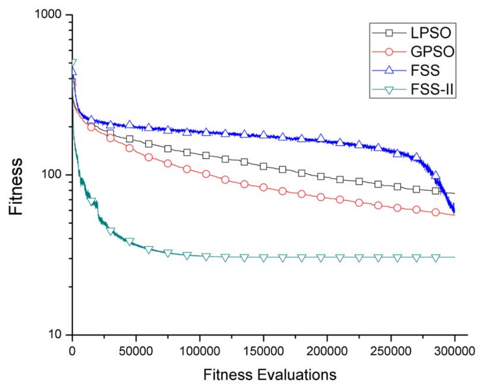

Fig. 3. Evolution of the fitness value for the Rastrigin function.

Figures 4, 5, 6, 7, 8 and 9 present the box plot for fitness value after $300,000$ fitness evaluations for the Sphere, Rosenbrock, Schwefel 1.2, Rastrigin, Griewank and Ackley functions, respectively. From the Figures, one can observe that our proposal far outperformed *FSS*, *GPSO* and *LPSO* in the Sphere, Schwefel 1.2 and Ackley functions. For the Rosenbrock function, the *FSS*-based approaches far outperformed the *PSO*-based approaches. For the Rastrigin function, *FSS-II* outperformed the other approaches, but there are some outliers.

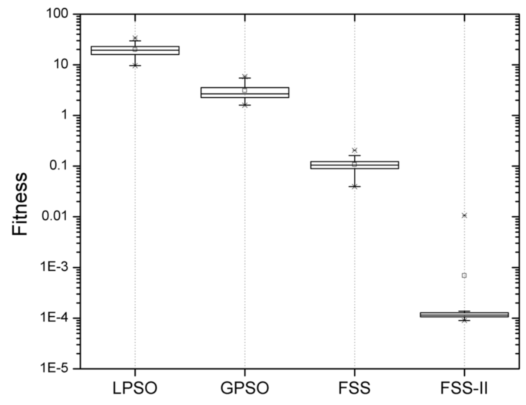

Fig. 4. Boxplot of the fitness value for the Sphere function.

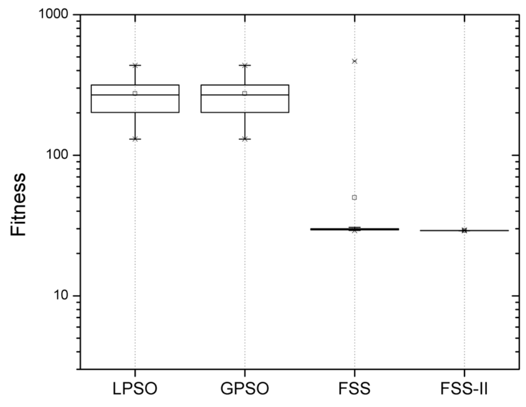

Fig. 5. Boxplot of the fitness value for the Rosenbrock function.

Since the results are similar for the Griewank function, we performed a non-parametrical statistic test. We used the Wilcoxon test with $p = 0.05$. The results are presented in Table II. One can observe that *FSS-II* outperformed all benchmark functions, except for the Griewank function. In this case, the *FSS* outperformed our approach, but we outperformed the *LPSO* and *FSS-II* is incomparable when compared to the *GPSO*.

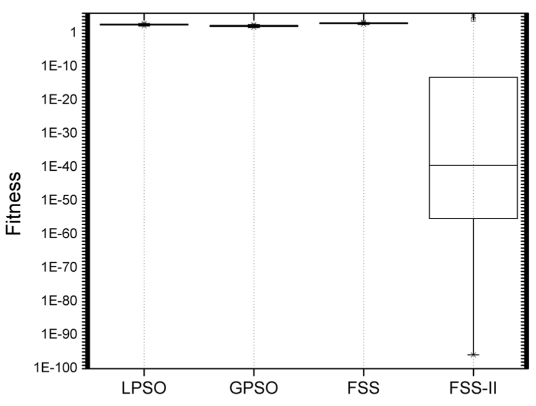

Fig. 6. Boxplot of the fitness value for the Schwefel 1.2 function.

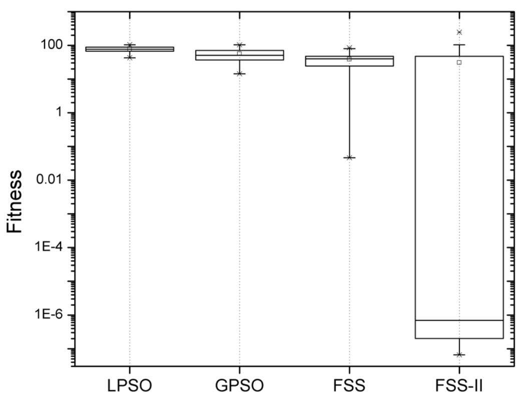

Fig. 7. Boxplot of the fitness value for the Rastrigin function.

We also analyzed the elapsed time to execute $300,000$ fitness evaluations in the Rastrigin function. The average time spent to perform all the 300.000 fitness evaluations, for the *FSS* and *FSS-II* are $69ms$ and $62.9ms$, respectively. We have observed that the *FSS-II* is faster than the *FSS*, even performing twice the number of iterations.

VI. Conclusions and Future Work

In this paper, we presented a novel version of the *FSS* algorithm, the *FSS-II* algorithm. Our proposal aims to mitigate three main drawbacks: low exploitation capability, necessity to evaluate the fitness function twice per iteration per fish and relative high complexity for implementation. The proposal is quite simple and just uses one global synchronization process, such as the global best in the *GPSO*. The global process is the barycenter evaluation. Since we are calculating the displacement based on the previous iterations to trigger the collective

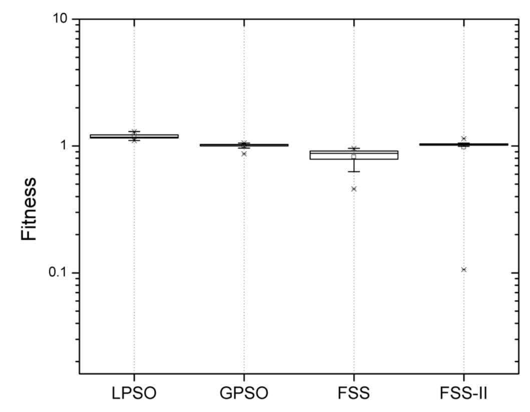

Fig. 8. Boxplot of the fitness value for the Griewank function.

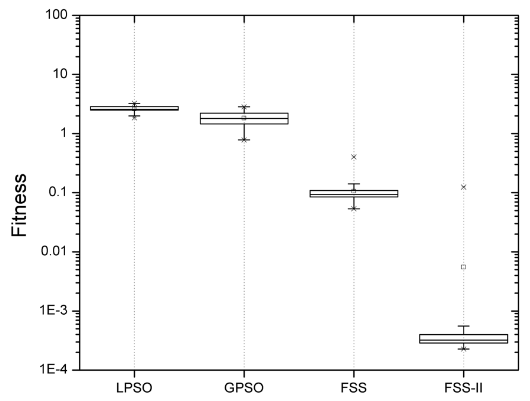

Fig. 9. Boxplot of the fitness value for the Ackley function.

movement, we just need to evaluate fitness function once per fish per iteration. Because of this, the novel algorithm is as simple as other widely used swarm intelligence algorithms.

We performed simulations in six well known benchmark function and our proposal outperformed the previous version of the *FSS* and the two most used *PSO* versions in most of cases.

We intend to perform a more detailed parametrical analysis and test the approach in high dimensional spaces with up to 500 dimensions.

TABLE II
RESULTS OF WILCOXON TEST FOR THE COMPARISON OF OUR PROPOSAL TO THE OTHER APPROACHES WITH $p = 0.05$.

FSSII	*FSS*	*GPSO*	*LPSO*
Sphere	▲	▲	▲
Rosenbrock	▲	▲	▲
Schwefel	▲	▲	▲
Rastrigin	▲	▲	▲
Griewank	▽	-	▲
Ackley	▲	▲	▲

REFERENCES

[1] M. Dorigo and G. Di Caro, "Ant colony optimization: A new meta-heuristic," in *Proceedings of the Congress on Evolutionary Computation*. IEEE Press, 1999, pp. 1470–1477.

[2] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey," in *Theoretical Computer Science*, vol. 344. Elsevier, 2005, pp. 243–278.

[3] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pp. 39–43, 1995.

[4] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[5] D. Bratton and J. Kennedy, "Defining a Standard for Particle Swarm Optimization," in *2007 IEEE Swarm Intelligence Symposium*, no. Sis. Ieee, Apr. 2007, pp. 120–127.

[6] B. Basturk and D. Karaboga, "An artificial bee colony (abc) algorithm for numeric function optimization," *IEEE Swarm Intelligence Symposium*, pp. 12–14, 2006.

[7] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Erciyes University, Engineering Faculty, Computer Engineering Department, Tech. Rep., 2005.

[8] Y. Tan and Y. Zhu, "Fireworks Algorithm for Optimization," *Advances in Swarm Intelligence*, vol. 46, no. 12, pp. 355–364, 2010.

[9] S. Zheng, A. Janecek, and Y. Tan, "Enhanced Fireworks Algorithm," in *2013 IEEE Congress on Evolutionary Computation*. Ieee, Jun. 2013, pp. 2069–2077.

[10] X. Yang and S. Deb, "Cuckoo search via Lévy flights," *World Congress on Nature & Biologically Inspired Computing, 2009. NaBIC 2009.*, pp. 210–214, 2009.

[11] J. Boelaert, "Kudu herd optimization," in *6th IEEE International Conference on Intelligent Systems (IS), 2012.*, 2012, pp. 3–8.

[12] Y. Tian, W. Gao, and S. Yan, "An Improved Inertia Weight Firefly Optimization Algorithm and Application," in *International Conference on Control Engineering and Communication Technology (ICCECT), 2012*. Ieee, Dec. 2012, pp. 64–68.

[13] C. J. A. Bastos-Filho, F. B. Lima-Neto, A. J. C. C. Lins, A. I. S. Nascimento, and M. P. Lima, "A novel search algorithm based on fish school behavior," *IEEE International Conference on Systems, Man, and Cybernetics*, pp. 2646–2651, 2008.

[14] C. J. A. Bastos-Filho, F. B. Lima-Neto, M. F. C. Sousa, M. R. Pontes, and S. S. Madeiro, "On the influence of the swimming operators in the fish school search algorithm," *2009 IEEE International Conference on Systems, Man and Cybernetics*, pp. 5012–5017, 2009.

[15] A. Janecek, "Feeding the Fish - Weight Update Strategies for the Fish School Search Algorithm," *Advances in Swarm Intelligence*, no. x, 2011. [Online]. Available: http://www.springerlink.com/index/Q28245837X648126.pdf

[16] L. T. Dos Santos, E. L. dos Santos Filho, and L. dos Santos Coelho, "Identificação e previsão de séries temporais utilizando LS-SVM otimizado pelo algoritmo de cardumes," *Learning and Nonlinear Models, Brazilian Computational Intelligence Society*, vol. 10, no. 2, pp. 119–126, 2012.

[17] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. Chen, A. Auger, and S. Tiwari, "Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization," Nanyang Technological University, Singapore and Kanpur Genetic Algorithms Laboratory, IIT Kanpur, India, Tech. Rep. 2005005, 2005. [Online]. Available: https://www.lri.fr/~hansen/Tech-Report-May-30-05.pdf