

A Hybrid Algorithm Based on Fish School Search and Particle Swarm Optimization for Dynamic Problems

George M. Cavalcanti-Júnior, Carmelo J.A. Bastos-Filho,
Fernando B. Lima-Neto, and Rodrigo M.C.S. Castro

Polytechnic School of Pernambuco, University of Pernambuco, Recife, Brazil
{gmcj,cjabf,fbln,rmcsc}@ecomp.poli.br

Abstract. Swarm Intelligence algorithms have been extensively applied to solve optimization problems. However, some of them, such as Particle Swarm Optimization, may not present the ability to generate diversity after environmental changes. In this paper we propose a hybrid algorithm to overcome this problem by applying a very interesting feature of the Fish School Search algorithm to the Particle Swarm Optimization algorithm, the collective volitive operator. We demonstrated that our proposal presents a better performance when compared to the FSS algorithm and some PSO variations in dynamic environments.

1 Introduction

The optima solutions for many real-world problems may vary over the time. For example, the optimal routes for a computer network can change dynamically due to nodes failures or due to unavailable links. Therefore, optimization algorithms to solve real-world problems should present the capability to deal with dynamic environments, in which the optima solutions can change along the time.

Many bio-inspired optimization algorithms have been proposed in the last two decades. Among them, there are the swarm intelligence algorithms, which were conceived based on some collective behaviors. In general, swarm algorithms are inspired in groups of animals, such as flocks of birds, schools of fish, hives of bees, colonies of ants, etc. Although a lot of swarm-based algorithms were already proposed, just some few were designed to tackle dynamic problems.

One of the most used swarm intelligence algorithms is the Particle Swarm Optimization (PSO). Despite the fast convergence capability, the vanilla version of the PSO can not tackle dynamic optimization problems. It occurs because the entire swarm often increases the exploitation around a good region of the search space, reducing the overall diversity of the population. However, some variations of the PSO have been created in order to increase the capacity to escape from regions in the search space where the optimum is not located anymore [1,2,3].

On the other hand, another swarm intelligence algorithm proposed in 2008, the Fish School Search algorithm (FSS) [4,5,6], presents a very interesting feature that can be very useful for dynamic environments. FSS presents an operator, called

volitive operator, which is capable to auto-regulate the exploration-exploitation trade-off during the algorithm execution.

Since the PSO algorithm converges faster than FSS but can not auto-adapt the granularity of the search, we believe the FSS volitive operator can be applied to the PSO in order to mitigate this PSO weakness and improve the performance of the PSO for dynamic optimization problems. Based on this, we propose in this paper a hybrid algorithm, called Volitive PSO.

This paper is organized as follows. Section 2 provides the background on PSO and FSS, also including a brief explanation of a well known PSO variation to tackle dynamic problems, called Charged PSO. Section 3 describes our proposal, which is a FSS-PSO hybrid algorithm. Section 4 presents the simulation setup. Section 5 is divided in two sub-sections and depicts some results. The former presents a parametrical analysis of our proposal and the latter shows a comparison between our proposal and some other approaches. In Section 6 we give our conclusions and we present some ideas for future works.

2 Background

2.1 PSO (Particle Swarm Optimization)

Particle Swarm Optimization is a population-based optimization algorithm inspired by the behavior of flocks of birds. It was firstly introduced by Kennedy and Eberhart [7] and it has been largely applied to solve optimization problems.

The standard approach is composed by a swarm of particles, where each one has a position within the search space \vec{x}_i and each position represents a solution for the problem. The particles fly through the search space of the problem searching for the best solution, according to the current velocity \vec{v}_i , the best position found by the particle itself (\vec{P}_{best_i}) and the best position found by the entire swarm during the search so far (\vec{G}_{best}).

According to the approach proposed by Shi and Eberhart [8] (this approach is also called inertia PSO), the velocity of a particle i is evaluated at each iteration of the algorithm by using the following equation:

$$\vec{v}_i(t+1) = w\vec{v}_i(t) + r_1c_1[\vec{P}_{best_i} - \vec{x}_i(t)] + r_2c_2[\vec{G}_{best} - \vec{x}_i(t)], \quad (1)$$

where r_1 and r_2 are numbers randomly generated in the interval $[0, 1]$. The inertia weight (w) controls the influence of the previous velocity and balances the exploration-exploitation behavior along the process. It generally decreases from 0.9 to 0.4 during the algorithm execution. c_1 and c_2 are called cognitive and social acceleration constants, respectively, and weights the influence of the memory of the particle and the information acquired from the neighborhood.

The position of each particle is updated based on the velocity of the particle, according to the following equation:

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1). \quad (2)$$

The communication topology defines the neighborhood of the particles and, as a consequence, the flow of information through the particles. There are two basic topologies: global and local. In the former, each particle shares and acquires information directly from all other particles, *i.e.* all particles use the same social memory, called G_{best} . In the local topology, each particle only share information with two neighbors and the social memory is not the same within the whole swarm. This approach, called L_{best} , helps to avoid a premature attraction of all particles to a single spot point in the search space.

2.2 Charged PSO

Since the standard PSO can not tackle dynamic problems due to the the low capacity to increase the diversity after the entire swarm has converged to a single region of the search space, many efforts to overcome this weakness have been made. The simplest idea is to restart the particles every time the search space changes. However, all the previous information obtained from the problem during the search process is lost in this case.

An interesting approach introduced by Blackwell and Bentley [1] is the Charged PSO, which uses the idea of electrostatic charges. Some particles are charged (they repeal themselves) and some others are neutral. In general, the neutral particles tend to exploit towards a single sub-region of the search space, whereas the charged particles never converges to a unique spot. Nevertheless, the charged particles are constantly exploring in order to maintain diversity.

In order to consider the effect of the charged particles, the velocity equation receives a fourth term, as shown in the equation (3). This term is defined as the acceleration of the particle i (\vec{a}_i) an can be seen in equation (4).

$$\vec{v}_i(t + 1) = w\vec{v}_i(t) + r_1c_1[\vec{P}_{best_i} - \vec{x}_i(t)] + r_2c_2[\vec{G}_{best} - \vec{x}_i(t)] + \vec{a}_i(t). \quad (3)$$

$$\vec{a}_i(t) = \begin{cases} \sum_{i \neq j} \frac{Q_i Q_j}{\|\vec{r}_{ij}(t)\|^3} \vec{r}_{ij}(t), & \text{if } R_c \leq \|\vec{r}_{ij}(t)\| \leq R_p, \\ 0, & \text{otherwise,} \end{cases} \quad (4)$$

where $\vec{r}_{ij}(t) = \vec{x}_i(t) - \vec{x}_j(t)$, Q_i is the charge magnitude of the particle i , R_c is the core radius and R_p is the perception limit of the particle. Neutral particles have charge value equal to zero, *i.e.* $Q_i = 0$.

2.3 FSS (Fish School Search)

The Fish School Search (FSS) is an optimization algorithm based on the gregarious behavior of oceanic fish. It was firstly proposed by Bastos-Filho *et al* in 2008 [4]. In the FSS, each fish represents a solution for the problem. The success of a fish during the search process is indicated by its weight. The FSS has four operators, which are executed for each fish of the school at each iteration: (i) individual movement, which is responsible for local search *stepind*; (ii) feeding, which updates the fish weights indicating the degree of success or failure during

the search process so far; (iii) collective-instinctive movement, which makes all fish moves toward a resultant direction; and (iv) collective-volitive movement, which controls the granularity of the search. In this paper, as we are dealing with dynamic environments, only the feeding and collective-volitive movement operators are used to build the proposed hybrid algorithm.

Feeding operator

The feeding operator determines the variation of the fish weight at each iteration. One should notice that a fish can increase or decrease its weight depending, respectively, on the success or failure during the search process. The weight of the fish is evaluated according to the following equation:

$$W_i(t+1) = W_i(t) + \frac{\Delta f_i}{\max(|\Delta f|)}, \quad (5)$$

where $W_i(t)$ is the weight of the fish i , Δf_i is the variation of the fitness function between the new position and the current position of the fish, $\max(|\Delta f|)$ is the absolute value of the greatest fitness variation among all fish. There is a parameter w_{scale} that limits the maximum weight of the fish. The weight of each fish can vary between 1 and w_{scale} and has an initial value equal to $\frac{w_{scale}}{2}$.

Collective-volitive movement operator

This operator controls the granularity of the search executed by the fish school. When the whole school is achieving better results, the operator approximates the fish aiming to accelerate the convergence toward a good region. On the contrary, the operator spreads the fish away from the barycenter of the school and the fish have more chances to escape from a local minimum. The fish school expansion or contraction is applied as a small drift to every fish position regarding the school barycenter, which can be evaluated as shown below:

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

We use equation (7) to perform the fish school expansion (use sign +) or contraction (use sign -).

$$\vec{x}_i(t+1) = \vec{x}_i(t) \pm step_{vol} r_1 \frac{\vec{x}_i(t) - \vec{B}(t)}{d(\vec{x}_i(t), \vec{B}(t))}, \quad (7)$$

where r_1 is a number randomly generated in the interval $[0, 1]$. $d(\vec{x}_i, \vec{B})$ evaluates the euclidean distance between the particle i and the barycenter. $step_{vol}$ is called volitive step and controls the step size of the fish. The $step_{vol}$ is bounded by two parameters ($step_{vol_min}$ and $step_{vol_max}$) and decreases linearly from $step_{vol_max}$ to $step_{vol_min}$ along the algorithm iterations. It helps the algorithm to initialize with an exploration behavior and change dynamically to an exploitation behavior.

3 Volitive PSO

This section presents the proposed algorithm, called Volitive PSO, which is a hybridization of the FSS and the PSO algorithms. Our proposal is to include two FSS operators in the Inertia PSO, the feeding and the collective-volitive movement. In the Volitive PSO, each particle becomes a weighted particle, where the weight is used to indicate the collective-volitive movement, resulting in expansion or contraction of the school. In our proposal, the $step_{vol}$ does not decrease linearly, it decreases according to equation (8). The parameter volitive step decay percentage ($decay_{vol}$) must be in the interval $[0, 100]$.

$$step_{vol}(t+1) = step_{vol}(t) \frac{100 - decay_{vol}}{100}. \quad (8)$$

The $step_{vol}$ is reinitialized to $step_{vol_max}$ when a change in the environment is detected. We use a sentry particle [9] to detect these changes. The fitness of the sentry particle is evaluated in the end of each iteration and in the beginning of the next iteration. The Algorithm 1.1 shows the Volitive PSO pseudocode.

Algorithm 1.1: Volitive PSO pseudocode

```

Initialize parameters and particles;
while the stop condition is not reached do
  foreach particle of the swarm do
    Evaluate the fitness of the particle;
    Evaluate  $\vec{P}_{best}$  and  $\vec{L}_{best}$ ;
  end
  if an environment change is detected then
    Initialize  $step_{vol}$ ;
  end
  foreach particle of the swarm do
    Update the velocity and the position of the particle;
    Evaluate the fitness of the particle;
  end
  Execute feeding operator;
  Execute collective-volitive movement operator;
  foreach particle of the swarm do
    Evaluate  $\vec{P}_{best}$  and  $\vec{L}_{best}$ ;
  end
  Update  $step_{vol}$  and  $w$ ;
end

```

4 Simulation Setup

In this section we present the benchmark function, the metric to measure the quality of the algorithms and the values for the parameters used in the simulations.

4.1 Benchmark Function

We used the DF1 benchmark function proposed by Morrison and Jong [10] in our simulations. DF1 is composed by a set of random peaks with different heights and slopes. The number of peaks, their heights, slopes, and positions within the search space are adjustable. The function for a N -dimensional space is defined according to the equation (9).

$$f(\vec{x}) = \max_{i=1,2,\dots,P} [H_i - S_i \sqrt{\sum (\vec{x} - \vec{x}_i)^2}], \quad (9)$$

where P is the number of peaks (peak i is centered in the position \vec{x}_i), H_i is the peak height and S_i is the peak slope. The values for x_{id} , H_i and S_i are bounded.

The dynamic components of the environment are updated using discrete steps. The DF1 uses a logistic function to control the generation of different step sizes. The parameter used to calculate the steps is adjusted according to the equation (10).

$$e_i = r e_{i-1} [1 - e_{i-1}], \quad (10)$$

where r is a constant in the interval $[1,4]$. As r increases, more simultaneous results for e are achieved. As r gets closer to 4, the behavior becomes chaotic.

The dynamics of the environment is specified using the following parameters: N_{peaks} is the number of peaks in motion; r_h is the r value for height dynamics; r_s is the r value for slope dynamics; r_{xd} is the r value for position dynamics in dimension d ; It is necessary to have a scaling factor for each r value.

4.2 Performance Metric

The mean fitness metric was introduced by Morrison [11]. He argued that a representative performance metric to measure the quality of an algorithm in a dynamic environment should reflect the performance of the algorithm across the entire range of environment dynamics. The mean fitness is the average over all previous fitness values, as defined below:

$$F_{mean}(T) = \frac{\sum_{t=1}^T F_{best}(t)}{T}, \quad (11)$$

where T is the total number of iterations and F_{best} is the fitness of the best particle after iteration t . The advantage of the mean fitness is that it represents the entire algorithm performance history.

We also used the collective mean fitness [11], that is simply the average value of the mean fitness at the last iteration over a predefined number of trials.

4.3 Parameters Settings

All results presented in this paper are the average values after 30 trials. We used 10,000 iterations for all algorithms. We performed the experiments in two situations: (i) 10 dimensions and 10 peaks and (ii) 30 dimensions and 30 peaks.

In this paper, only the peak positions are varied along the iterations. The heights and slopes of the peaks were initialized randomly within the predefined interval. The parameters used for the DF1 function are $H_{base} = 40$, $H_{range} = 20$, $H_{scale} = 0.5$, $r_h = 3.2$, $S_{base} = 1$, $S_{range} = 7$, $S_{scale} = 0.5$, $r_s = 1.2$, $x_{base_id} = -10$, $x_{range_id} = 20$, $x_{scale_id} = 0.7$, $r_{xd} = 3.2$.

For all PSO algorithms, we used 50 particles, local topology, c_1 and c_2 equal to 1.494 [12] and w decreasing linearly from 0.9 to 0.4 along 100 iterations. We set up $w = 0.9$ every time an environment change is detected. We chose the local topology since it helps to avoid premature convergence to a local optimum, which is good for optimization in dynamic environments. The Charged PSO was tested empirically with 30%, 50% and 70% of charged particles, and for $Q = 4$, $Q = 8$, $Q = 12$ and $Q = 16$. In both scenarios, the best results were achieved for 30% of charged particles and $Q = 12$. Hence, these values were used. For the FSS, we used 50 fish, $W_{scale} = 500$, initial and final individual step equal to 2% and 0.01%, and initial and final volitive step equal to 40% and 0.1%. $step_{ind}$ and $step_{vol}$ decreases linearly along 100 iterations and are reinitialized when a change in environment occurs. For the Volitive PSO, we used $w_{scale} = 500$, and $step_{vol_min} = 0.01\%$.

5 Results

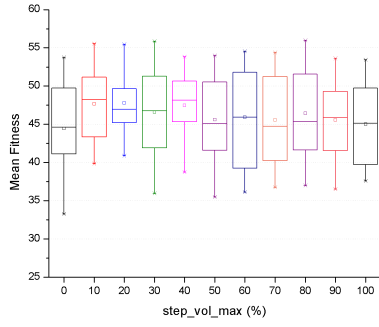
5.1 Analysis of the Parameters

This section presents an analysis of the influence of the parameters $decay_{vol}$ and $step_{vol_max}$ in the performance of the Volitive PSO. As preliminary results showed that the algorithm is more sensible to the $decay_{vol}$ parameter and high values for $decay_{vol}$ do not present good performance, we tested the following $decay_{vol}$ values: 0%, 10% and 25%. For each $decay_{vol}$ value, we varied the $step_{vol_max}$ value and the box plots of the mean fitness at the last iteration are shown in the Figure 1.

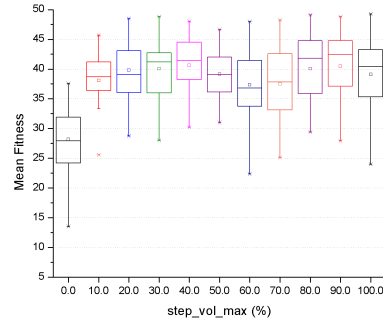
For the case 1 (10 dimensions and 10 peaks), the average mean fitness for different $step_{vol_max}$ are not so different (as shown in Figures 1(a), 1(c) and 1(e)). However, slightly better results can be observed for $decay_{vol} = 10\%$. Nevertheless, for the case 2 (30 dimensions and 30 peaks), the best results were achieved for $decay_{vol}$ equal to 0%. It indicates that is better to not diminish the $step_{vol}$ for spaces with higher dimensionality. The best results for the case 2 were achieved when $step_{vol_max} = 40\%$ and $decay_{vol} = 0\%$. Hence, we used these values for the comparison presented in the next sub-section.

5.2 Comparison with Other Approaches

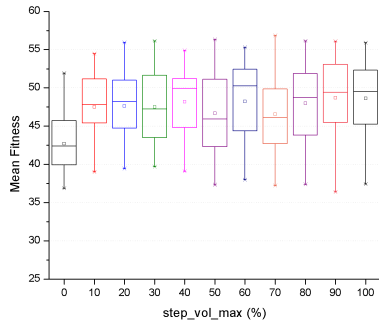
In this section we present a brief performance comparison among the Volitive PSO, Inertia PSO, Restart PSO (simply reinitialize the particles when a change in the environment is detected), Charged PSO and FSS. Figure 2 depicts the average values of fitness for each algorithm. As can be seen, the Volitive PSO achieved better results in average than the other algorithms in both cases.



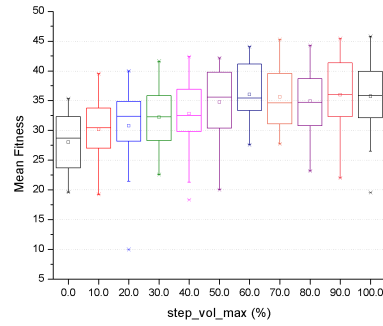
(a) $decay_{vol} = 0\%$, 10d and 10 peaks.



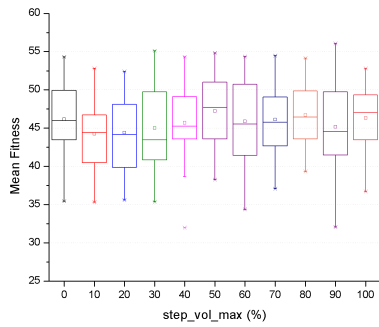
(b) $decay_{vol} = 0\%$, 30d and 30 peaks.



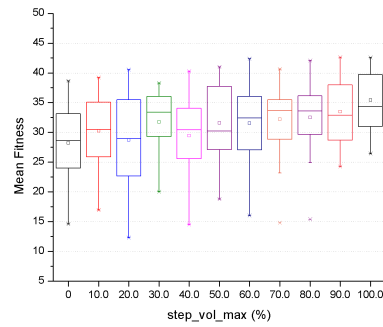
(c) $decay_{vol} = 10\%$, 10d and 10 peaks.



(d) $decay_{vol} = 10\%$, 30d and 30 peaks.

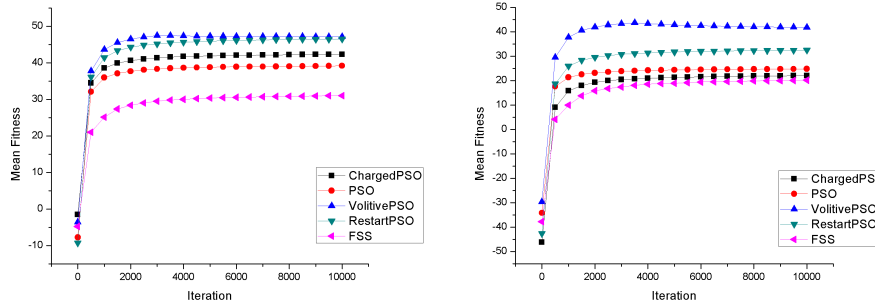


(e) $decay_{vol} = 25\%$, 10d and 10 peaks.



(f) $decay_{vol} = 25\%$, 30d and 30 peaks.

Fig. 1. Analysis of the parameters $decay_{vol}$ and $step_{vol_max}$ of the Volitive PSO algorithm



(a) 10 dimensions and 10 peaks. (b) 30 dimensions and 30 peaks.

Fig. 2. Comparative evolution of the algorithms on the DF1 function

Table 1. Collective Mean Fitness - Average (standard deviation) after 10,000 iterations

(a) 10 dimensions and 10 peaks. (b) 30 dimensions and 30 peaks.

PSO	39.207 (6.533)	PSO	24.827 (5.486)
Restart PSO	46.528 (5.590)	Restart PSO	32.493 (6.088)
Charged PSO	42.249 (4.542)	Charged PSO	22.039 (6.965)
FSS	31.032 (9.742)	FSS	20.192 (7.340)
Volitive PSO	47.168 (4.517)	Volitive PSO	41.854 (4.521)

Table 1 shows the collective mean fitness (and standard deviation in parenthesis) after 10,000 iterations. One can observe that the Volitive PSO also achieved lower standard deviation in both cases.

6 Conclusion

In this paper we proposed a hybrid FSS-PSO algorithm for dynamic optimization. We showed that the collective-volitive movement operator applied to the PSO can help to maintain diversity when the search space is varying over the time, without reducing the exploitation capability. Some preliminary results showed that the volitive step must not decay quickly. It indicates the important role of the FSS-operator to generate diversity after environmental changes. Further research includes a deeper analysis of the Volitive PSO and more tests varying the peaks height and slopes. Also, we intend to analyze the dynamics of the swarm within the search space.

Acknowledgments

The authors acknowledge the financial support from CAPES, CNPq and University of Pernambuco for scholarships, support and travel grants.

References

1. Blackwell, T.M., Bentley, P.J.: Dynamic Search with Charged Swarms. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 19–26 (2002)
2. Rakitianskaia, A., Engelbrecht, A.P.: Cooperative charged particle swarm optimiser. In: Congress on Evolutionary Computation, CEC 2008, pp. 933–939 (June 2008)
3. Nickabadi, A., Ebadzadeh, M.M., Safabakhsh, R.: Evaluating the performance of DNPSO in dynamic environments. In: IEEE International Conference on Systems, Man and Cybernetics, pp. 2640–2645 (October 2008)
4. Bastos-Filho, C.J.A., Neto, F.B.L., Lins, A.J.C.C., Nascimento, A.I.S., Lima, M.P.: A novel search algorithm based on fish school behavior. In: IEEE International Conference on Systems, Man and Cybernetics, pp. 2646–2651. IEEE, Los Alamitos (October 2009)
5. Bastos-Filho, C.J.A., Neto, F.B.L., Sousa, M.F.C., Pontes, M.R.: On the Influence of the Swimming Operators in the Fish School Search Algorithm. In: SMC, pp. 5012–5017 (October 2009)
6. Bastos-Filho, C.J.A., de Lima Neto, F.B., Lins, A.J.C.C., Nascimento, A.I.S., Lima, M.P.: Fish school search. In: Chiong, R. (ed.) Nature-Inspired Algorithms for Optimisation. SCI, vol. 193, pp. 261–277. Springer, Heidelberg (2009)
7. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks, vol. 4, pp. 1942–1948 (1995)
8. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: The 1998 IEEE International Conference on Evolutionary Computation Proceedings, IEEE World Congress on Computational Intelligence, pp. 69–73 (1998)
9. Carlisle, A., Dozier, G.: Applying the particle swarm optimizer to non-stationary environments. Phd thesis, Auburn University, Auburn, AL (2002)
10. Morrison, R.W., Jong, K.A.D.: A test problem generator for non-stationary environments. In: Proc. of the 1999 Congr. on Evol. Comput., pp. 2047–2053 (1999)
11. Morrison, R.W.: Performance Measurement in Dynamic Environments. In: GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems, pp. 5–8 (2003)
12. Eberhart, R.C., Shi, Y.: Particle Swarm Optimization: Developments, Applications and Resources. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2001 (2001)