We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900
Open access books available

185,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



5

Volitive Clan PSO - An Approach for Dynamic Optimization Combining Particle Swarm Optimization and Fish School Search

George M. Cavalcanti-Júnior, Carmelo J. A. Bastos-Filho
and Fernando B. de Lima-Neto
Polytechnic School of Pernambuco, University of Pernambuco
Brazil

1. Introduction

The optima solutions for many real-world problems may vary over the time. Therefore, optimization algorithms to solve this type of problem should present the capability to deal with dynamic environments, in which the optima solutions can change during the algorithm execution. Many swarm intelligence algorithms have been proposed in the last years, and in general, they 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 of them 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 standard version of the *PSO* can not tackle dynamic optimization problems. It occurs mainly because the entire swarm often increases the exploitation around a good region of the search space, consequently reducing the overall diversity of the population. Some variations of the *PSO*, such as *Charged PSO* proposed by Blackwell & Bentley (2002) and *Heterogeneous PSO* proposed by Leonard et al. (2011), have been proposed in order to increase the capacity to escape from regions within the search space where the global optimum is not located anymore.

The topology of the *PSO* defines the communication schema among the particles and it plays an important hole in the performance of the algorithm. The topology can influence in the trade-off between the convergence velocity and the quality of the solutions. In general, *PSO* topologies that benefit diversity, *e.g.* local and Von Neumann, are used to handle dynamic optimization problems. Carvalho and Bastos-Filho (Carvalho & Bastos-Filho, 2009a) presented a dynamic topology based on clan behaviors, which improves the *PSO* performance in various benchmark functions. This approach was named *Clan PSO*.

On the other hand, another swarm intelligence algorithm proposed in 2008, called the Fish School Search algorithm (*FSS*) (Bastos Filho, de Lima Neto, Lins, Nascimento & Lima, 2009), presents a very interesting feature that can be very useful for dynamic environments. *FSS* has an operator, called collective-volitive, which is capable to self-regulate automatically the exploration-exploitation trade-off during the algorithm execution.

Since the *PSO algorithm* converges faster than the *FSS*, but can not self-adapt the granularity of the search, Cavalcanti-Júnior *et al.* (Cavalcanti-Júnior et al., 2011) have incorporated the *FSS* volitive operator into the *PSO* in order to allow diversity generation after an stagnation process. The algorithm was named *Volitive PSO*. On dynamic optimization benchmark functions, this algorithm obtained better results than some *PSO* approaches created to tackle dynamic optimization problems, such as the *Charged PSO* (Blackwell & Bentley, 2002).

We believe that one can profit better results in multimodal search spaces if we deploy a collaborative multi-swarm approach in the *Volitive PSO*. Thus, we propose in this chapter to run independently the volitive operator in each one of the clans of the *Clan PSO* algorithm.

The chapter is organized as follows: Sections 2, 3, 4 and 5 present the background on *PSO*, *Clan PSO*, *FSS* and *Volitive PSO*, respectively. In Section 6 we put forward our contribution, the *Volitive Clan PSO*. In Section 7 we present the simulation setup. In Sections 8 we analyze the dynamics of our proposal and compare it to previous approaches. In Section 9.1 and 9.2 we present, respectively, some simulation results regarding the dependence on the parameters and compare our proposal to previous approaches in terms of performance. In Section 10 we give our conclusions.

2. *PSO* (Particle Swarm Optimization)

PSO is a population-based optimization algorithm inspired by the behavior of flocks of birds. 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 possible 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 neighborhood of the particle i during the search so far ($\vec{N}_{best_i}$). One of the most used approach was proposed by Shi and Eberhart (Shi & Eberhart, 1998). This approach is also called *Inertia PSO*. According to their approach, the velocity of a particle i is evaluated at each iteration of the algorithm by using the following equation:

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

where r_1 and r_2 are numbers randomly generated in the interval $[0, 1]$ by an uniform probability density function. The inertia weight (w) controls the influence of the previous velocity and balances the exploration-exploitation behavior along the process. c_1 and c_2 are called cognitive and social acceleration constants, respectively, and weight 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 updated 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 (Figure 1(a)) and local (Figure 1(b)). In the former, each particle shares and acquires information directly from all other particles, *i.e.* all particles use the same social memory, often referred as $\vec{G}_{best}$. 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, often called $\vec{L}_{best}$, helps to avoid a premature attraction of all particles to a single spot point in the search space, but presents a slower convergence.

Many other topologies were already proposed to overcome this trade-off. One promising topology, called *Clan PSO*, has a dynamic structure and outperforms the standard topologies in multimodal search spaces.



(a) Global best topology. (b) Local best topology.

Fig. 1. Two basic topologies used in PSO.

3. *Clan PSO*

Clans are groups of individuals united by a kinship and each clan has at least one guide. Inspired by these leadership characteristics, Carvalho and Bastos-Filho (Carvalho & Bastos-Filho, 2009a) proposed a topology called *Clan PSO*. As a part of the topology, each clan is composed by particles which are connected to the other particles of the same clan in order to share information quickly, as shown in the example depicted in Figure 2. The algorithm

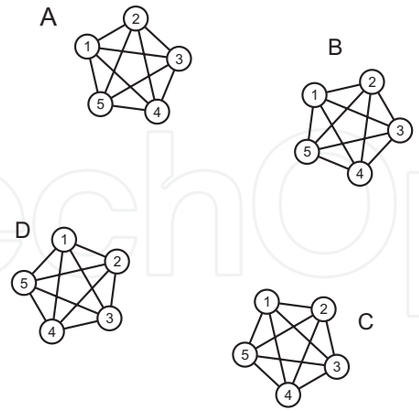


Fig. 2. Topology used within the clans for the *Clan PSO* with 4 clans with 5 particles.

has two phases executed in every iteration, the delegation of the leaders and the conference of the leaders. In previous works, Carvalho and Bastos-Filho (Carvalho & Bastos-Filho, 2009a) recommended to use just some few clans in order to avoid extra overhead within the iteration. More details about them are given in the following subsections.

3.1 Delegation of the leaders

At each iteration, each clan singly performs an independent *PSO* and the particle that obtained the best position within the clan is delegated as a leader of the clan in the iteration. Figure 3 illustrates an example with the leader delegated in each clan.

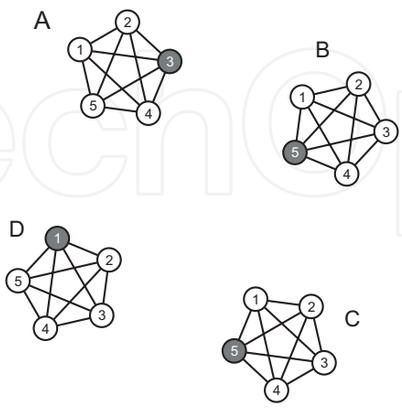
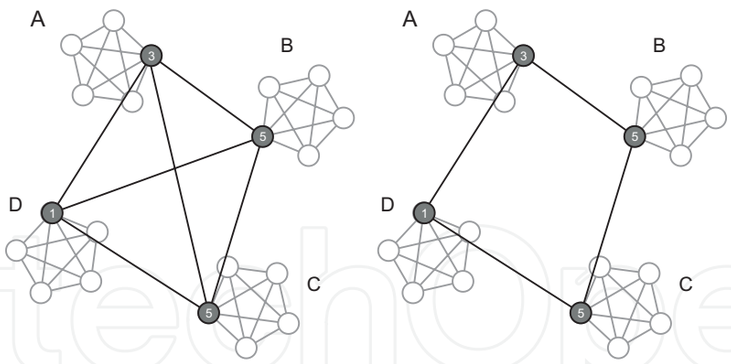


Fig. 3. Leaders delegated in the clans (A, B, C, D).

3.2 Conference of the leaders

After the Delegation, the leaders of each clan are selected and a new virtual swarm is composed by them. A *PSO* with the leaders can be ran using global (Figure 4(a)) or local (Figure 4(b)) topology. The former induces a faster convergence, while the latter allows more exploitation.



(a) Global conference. (b) Local conference.

Fig. 4. Leaders conference illustrations.

We will use the following nomenclature along the paper: <number of clans> x <particles per clan>. This means that if the swarm configuration is 4x5 particles, then the whole swarm contains 4 clans and each clan has 5 particles.

4. *FSS* (Fish School Search)

The Fish School Search (*FSS*) is an optimization algorithm based on the gregarious behavior of schools. It was firstly proposed by Bastos-Filho *et al.* (Bastos Filho, de Lima Neto, Lins,

Nascimento & Lima, 2009). In the *FSS*, each fish represents a solution for the problem and 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; (ii) feeding; (iii) collective-instinctive movement; and (iv) collective-volitive movement. Since we will use only the feeding and collective-volitive movement operators in our proposal, we detail them in the following subsections.

4.1 Feeding operator

The feeding operator determines the variation of the fish weight at each iteration. 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 equation (3):

$$W_i(t+1) = W_i(t) + \frac{\Delta f_i}{max(|\Delta f|)}, \tag{3}$$

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 highest fitness variation among all fish in the current iteration. 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 $\frac{w_{scale}}{2}$.

4.2 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 school has more chances to escape from a local optimum. The fish school expansion or contraction is applied as a small drift to every fish position regarding the school barycenter, which can be evaluated by using the equation (4):

$$\vec{B}(t) = \frac{\sum_{i=1}^{N} \vec{x}_i(t) W_i(t)}{\sum_{i=1}^{N} \vec{x}_i(t)}. \tag{4}$$

We use equation (5) to perform the fish school expansion (in this case we use sign $+$) or contraction (in this case we use sign $-$).

$$\vec{x}_i(t+1) = \vec{x}_i(t) \pm step_{vol} r_1 \frac{\vec{x}_i(t) - \vec{B}(t)}{d(\vec{x}_i(t), \vec{B}(t))}, \tag{5}$$

where r_1 is a number randomly generated in the interval $[0, 1]$ by an uniform probability density function. $d(\vec{x}_i(t), \vec{B}(t))$ 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 and is defined as a percentage of the search space range. 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 iterations of the algorithm. It helps the algorithm to initialize with an exploration behavior and change dynamically to an exploitation behavior.

5. *Volitive PSO*

Volitive PSO is a hybridization of the *FSS* and the *PSO* algorithms and it was proposed by (Cavalcanti-Júnior et al., 2011). The algorithm uses two *FSS* operators in the *Inertia PSO*, the feeding and the collective-volitive movement. Each particle becomes a weighted particle, where the weights are used to define the collective-volitive movement, resulting in an expansion or contraction of the school. As a results, the *Volitive PSO* presents good features of the *PSO* and the *FSS* to tackle dynamic problems. These features are, respectively, fast convergence and the capacity to self-regulate the granularity of the search by using the volitive operator. Figure 5 illustrates the features aggregated in the *Volitive PSO*.

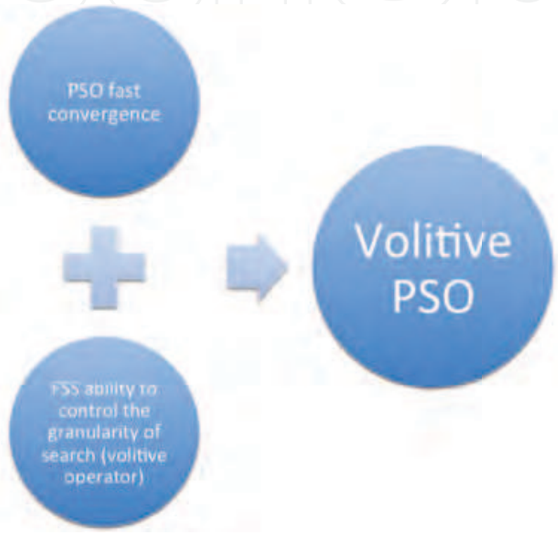


Fig. 5. Illustration of the features presented by the *Volitive PSO*.

In the *Volitive PSO*, the $step_{vol}$ decreases according to the equation (6).

$$step_{vol}(t+1) = step_{vol}(t)\frac{100 - decay_{vol}}{100}, \tag{6}$$

where $decay_{vol}$ is the volitive step decay percentage which must be in the interval $[0, 100]$.

The $step_{vol}$ is reinitialized to $step_{vol_max}$ when a change in the environment is detected in order to allow the algorithm to increase the diversity within the population. The detection of a change is performed by using a sentry particle as presented in (Carlisle & Dozier, 2002). In this case, the fitness is evaluated in the end of each iteration and in the beginning of the next iteration. Thus, immediately after an environment change, the algorithm has more capacity to escape from an old optima due to the larger steps of the volitive movement. As $step_{vol}$ is decreased along the iterations, the algorithm gradually changes from exploration to exploitation mode.

6. Our proposal: *Volitive clan PSO*

We propose here to include the volitive operator into the *Clan PSO*. In our approach, each clan runs a Volitive PSO separately and the weights of the individuals of each clan are treated independently. As a consequence, each clan can perform independent volitive movements,

shrinking or expanding its radius depending on its success or failure. The multi-swarm proposal with the volitive movement is the main difference to the approach proposed by Cavalcanti-Júnior et al. (2011). The pseudocode of our proposal is depicted in algorithm 1. We observed that the *Volitive Clan PSO* returned better results for dynamic environments when we run *PSOs* with local topology either within the clans and in the conference of the leaders.

Algorithm 1: Pseudocode of the *Volitive Clan PSO*.

```
Initialize parameters and particles;
while the stop condition is not reached do
    Update the sentry particle;
    if the sentry particle detected a change then
        reinitialize step_vol;
    foreach clan of the swarm do
        foreach particle of the clan do
            Update the velocity and the position of the particle using local topology
            using equations (1) and (2), respectively;
        Execute the feeding operator in the clan using equation (3);
        Execute the collective-volitive movement operator in the clan using equation
        (5);
        foreach particle of the clan do
            Evaluate the fitness of the particle;
            Evaluate P⃗_best and N⃗_best;
        Delegate the leader of the clan;
    Perform the conference of the Leaders using local topology;
    Update the sentry particle;
```

7. Simulation setup

All experiments were developed in JAVA and executed in a computer with a 2.40GHz Core 2 Quad processor and 8GB RAM memory running Linux operational system.

7.1 Benchmark function

We used the DF1 benchmark function proposed by Morrison & De Jong (1999) 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. As those three components can change during the execution, then they are called dynamic components. The function for a N-dimensional space is defined according to the equation (7).

$$f(\vec{x}) = max_{i=1,2,...,P}[H_i - S_i\sqrt{\sum(\vec{x} - \vec{x_i})^2}], \tag{7}$$

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 logistics function to control the generation of different step sizes. The parameter used to

calculate the steps is adjusted according to the equation (8).

$$e_i = re_{i-1}(1 - e_{i-1}), \tag{8}$$

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.

In the simulations presented in the Sections 9.1 and 9.2, we use 10 dimensions, 10 or 50 peaks and search space bounds $[-50, 50]^d$ for all dimensions. All peaks have a constant slope along the execution that is determined in the beginning of each simulation randomly in the interval $[1, 7]$. All peaks move around the bounded search space independently and their height vary in the interval $[10, 50]$, both for each 100 iterations. For Section 8 we use a search space with 2 dimensions.

We simulate an environment with high severity changes. For all dynamic components the scale parameter was set to 0.5. This parameters is a value between 0 and 1, which multiplies the result of the logistics function for each environment change. Thus, the scale parameter control the severity of the change of each dynamic component. The coefficient r of the logistic function is equal to 2.1 for all dynamic components.

All environments are generated using the same seed for the random number. Thus, the initial environment conditions are the same for all simulations. However, the dynamics of the environment over the algorithm execution are different for each simulation. For the box plot graphs, we evaluate the performance of the algorithms over 30 independent simulations with 10,000 iterations each one.

7.2 Performance metric

To measure the performance of an optimization algorithm in a dynamic environment a good metric should reflect the performance of the algorithm across the entire range of environment dynamics. Accordingly, we use in all experiments the mean fitness, which was introduced by Morrison (Morrison, 2003). The mean fitness is the average over all previous fitness values, as defined below:

$$F_{mean}(T) = \frac{\sum_{t=1}^{T} F_{best}(t)}{T}, \tag{9}$$

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.

7.3 Algorithms setup

The cognitive acceleration coefficient (c_1) of the *PSO* is set initially to 2.5 and decreases linearly to 0.5 along 100 iterations (that corresponds to the frequency of the environment change). On the other hand, the social acceleration coefficient (c_2) is initially equal to 0.5 and increases linearly to 2.5 over the same change interval. For every 100 iterations, c_1 and c_2 are reinitialized. Thus, the algorithms has more capacity to generate diversity after an environment change and consequently is more capable to explore the search space looking for new optima. Gradually, the algorithm increases the exploitation until another environment change occurs, then it return to the first step of this process. We used the inertia weight (w) equal to 0.729844 and a total number of 54 particles for all algorithms.

On the *Charged PSO*, c_1 and c_2 are constant and equal to 1.49618. 50% of the particles are charged with charge value 16, both according to the specification presented in (Blackwell & Bentley, 2002). The parameters p and p_{core} are set to 1 and 30, respectively, according to Blackwell Leonard et al. (2011).

For the parameters in *Clan PSO*, *Volitive PSO* and *Volitive Clan PSO*, we use the same configuration used in the *PSO*. For the algorithms which use the volitive operator, we use $w_{scale} = 5000$ and $step_{vol_min} = 0.01\%$, according to Bastos-Filho, Lima-Neto, Sousa & Pontes (2009).

8. Analysis of the dynamics of the algorithms

The following requirements are necessary to reach good performance in dynamic optimization: i) generation of diversity to explore the search space after an environment change, and ii) quick convergence to a new optimum. These capabilities can lead the algorithm to track optima solutions. In this section we analyze the dynamic behavior of our proposal and compare it to some other previous PSO approaches.

Figures 6, 7 and 8 present the positions of the particles for the *PSO* using Local topology with 54 particles (*PSO-L*), *Clan PSO* with 3 clans and 18 particles per clan (*ClanPSO-L 3x18*) and *Volitive Clan PSO* with 3 clans and 18 particles per clan (*Volitive Clan PSO-L 3x18*), respectively, for the two dimensional dynamic DF1 function (Morrison & De Jong, 1999). In the clan-based approaches, we used Local topology in the conference of leaders. In this analysis, we used the Global topology within the clans for the *Clan PSO* and the Local topology within the clans for the *Volitive Clan PSO*. All algorithms are deployed to maximize the DF1 function, where each red region represents a peak which changes its height and position after 100 iterations. The value for the peak height can be inferred by the legend situated on the right side of each graph. All figures show the positions of the particles: (a) just before an environment change, (b) just after an environment change and (c) 10, (d) 30, (e) 50 and (f) 100 iterations after the environment change.

From Figure 6, it is possible to observe that the *PSO-L* swarm is located in an outdated optimum position in the first iterations after the change in the environment. Because *PSO-L* does not have any mechanism to generate diversity after the swarm convergence, the swarm slows down to find another optimum, and just can find it because of the inertia term and reinitialization of c_1 and c_2. Figure 6(d) shows that most of the particles is located at an optimum which is not the global one after 30 iterations. One can observe that some particles escaped to other optima after 50 and 100 iterations, as shown in Figure 6(e) and 6(f). Nevertheless, the swarm could not generate diversity enough to explore farther regions of the search space in order to find other optima which could be the global one.

The *ClanPSO-L* presents a slightly worse behavior when compared to the *PSO-L*. Figure 7(a) shows that the swarm converged to a single spot and, after the environment change (Figure 7(b)), the whole swarm tends to move towards the optimum which is closest to this spot (Figures 7(c), 7(d)). Even after more iterations after the environment change, the swarm was not capable to generate diversity and, as a consequence, was not able to find an optimum far from the initial spot, as shown in Figures 7(e) and 7(f). This behavior occurs because every clan uses global topology, which strong attracts the whole sub-swarm to a single spot.

Unlike the *PSO-L* and the *ClanPSO-L*, the *Volitive Clan PSO* has a mechanism to generate diversity after an environment change. Comparing Figures 8(a) and 8(b), we can observe that the swarm spreads away from the barycenter after the change. It occurs because immediately after the change, the particles assess their positions and check that they are in a worse position than the last iteration because of the change in the environment. Because of this, the swarm tends to decrease its weight according to the feeding operator and it consequently triggers the collective-volitive operator to expand the swarm radius by repelling particles from the barycenter (Figure 8(b)). In Figures 8(c) and 8(d) one can observe that the swarm is still performing exploration, but the particles also begin to approximate themselves in order to converge to another optimum. Finally, in Figure 8(f) the swarm splits in three sub-swarms and each one is located in a different optimum. We believe that each sub-swarm is a clan since we are using the 3x18 configuration. Summarizing, Figure 8 shows that the *Volitive Clan PSO* is capable to generate diversity in order to escape from outdated optima.

9. Simulation results

9.1 Parametric analysis

In this section we analyse the impact of some parameters on the performance of the *Volitive Clan PSO*. We tested the following values for $decay_{vol}$: 0%, 5% and 10%, combined with $step_{vol_max}$ values equals to 30%, 40%, 50%, 60%, 70% and 80% of the search space range. We observed that the best results for the *Volitive Clan PSO* were achieved with Local topology within the clans and in the conference of the leaders. Therefore, we used these configurations in all experiments presented in this section.

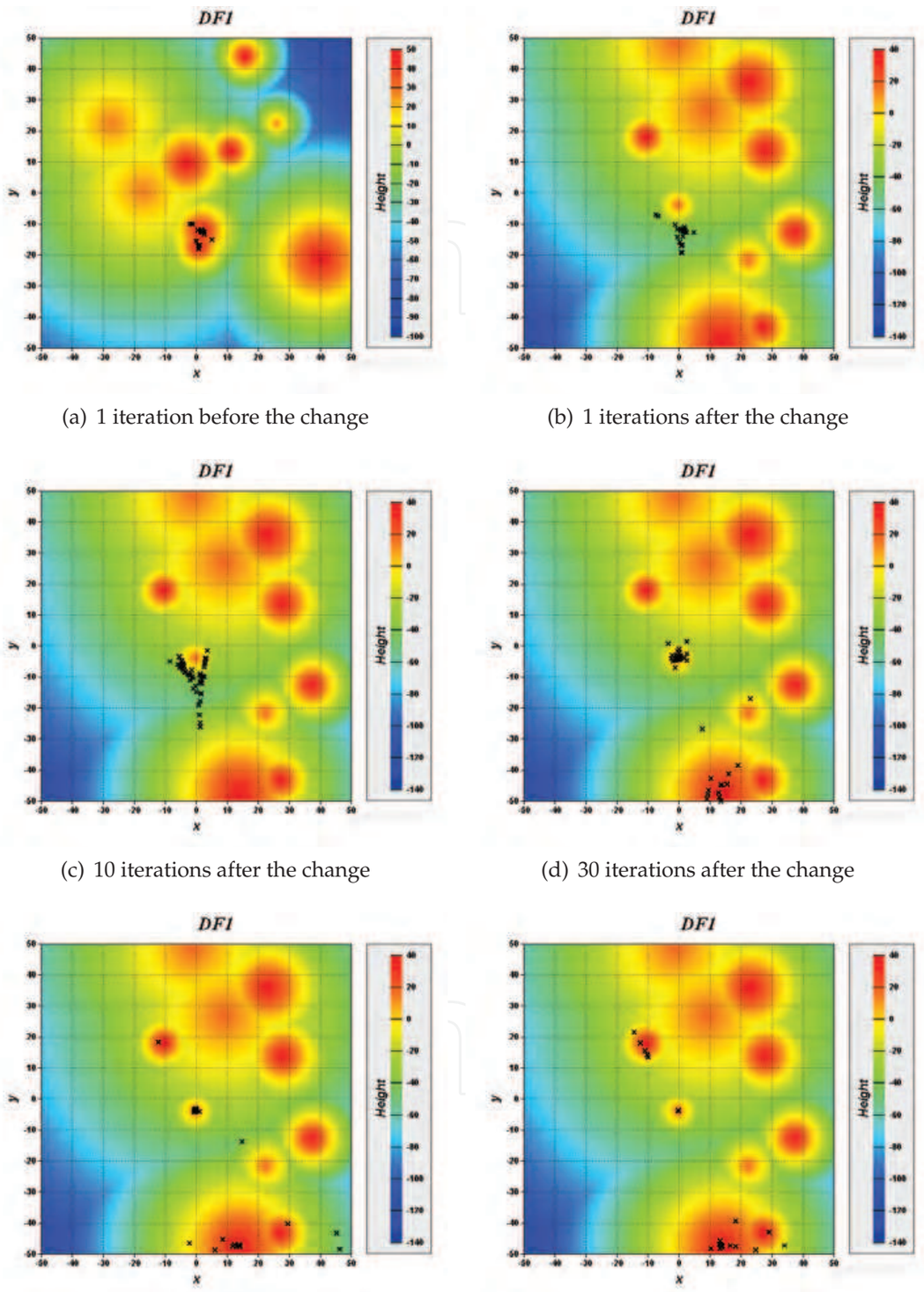
Figure 9 provides the performance for different values of $decay_{vol}$ and $step_{vol_max}$ for the configuration 3x18. The bests results were achieved for $decay_{vol} = 5\%$, as shown in Figure 9(b). One can observe that it is necessary to balance the $decay_{vol}$ value. If $decay_{vol} = 0\%$, then the algorithm is not allowed to exploit. On the other hand, if $decay_{vol}$ is higher, then $step_{vol}$ decays too fast and causes a premature convergence of the swarm.

According to results showed in Figure 9, we selected $decay_{vol} = 5\%$ and $step_{vol_max} = 60\%$ to compare different configurations of clans. We assessed the following clans configurations: 1x54, 3x18, 6x9 and 9x6. Figure 10 shows the results. One can observe that the best performance was achieved for 3x18 particles. Therefore, we used this configuration to compare the performance to other algorithms (experiments presented in the Section 9.2).

9.2 Performance comparison

We compare all algorithms in two situations: without reinitializing particles and reinitializing 50% of particles for every environment change. The second situation is a common approach to generate diversity in algorithms when dealing with dynamic problems (Leonard et al., 2011). In both situations we performed simulations with 10 and 50 peaks for the DF1 benchmark function.

Figure 11 shows the box plots for the performance of the algorithms in terms of Mean Fitness for 10 dimensions and 10 peaks. Comparing the Figures 11(a) and 11(b) we can observe that the algorithms that uses the volitive operator achieved better results in both situations. Besides, the results observed for the *Volitive PSO* and the *Volitive Clan PSO* did not change significantly when the reinitialization procedure is used (see Tables 0(a) and 0(b)). In fact, the



(a) 1 iteration before the change

(b) 1 iterations after the change

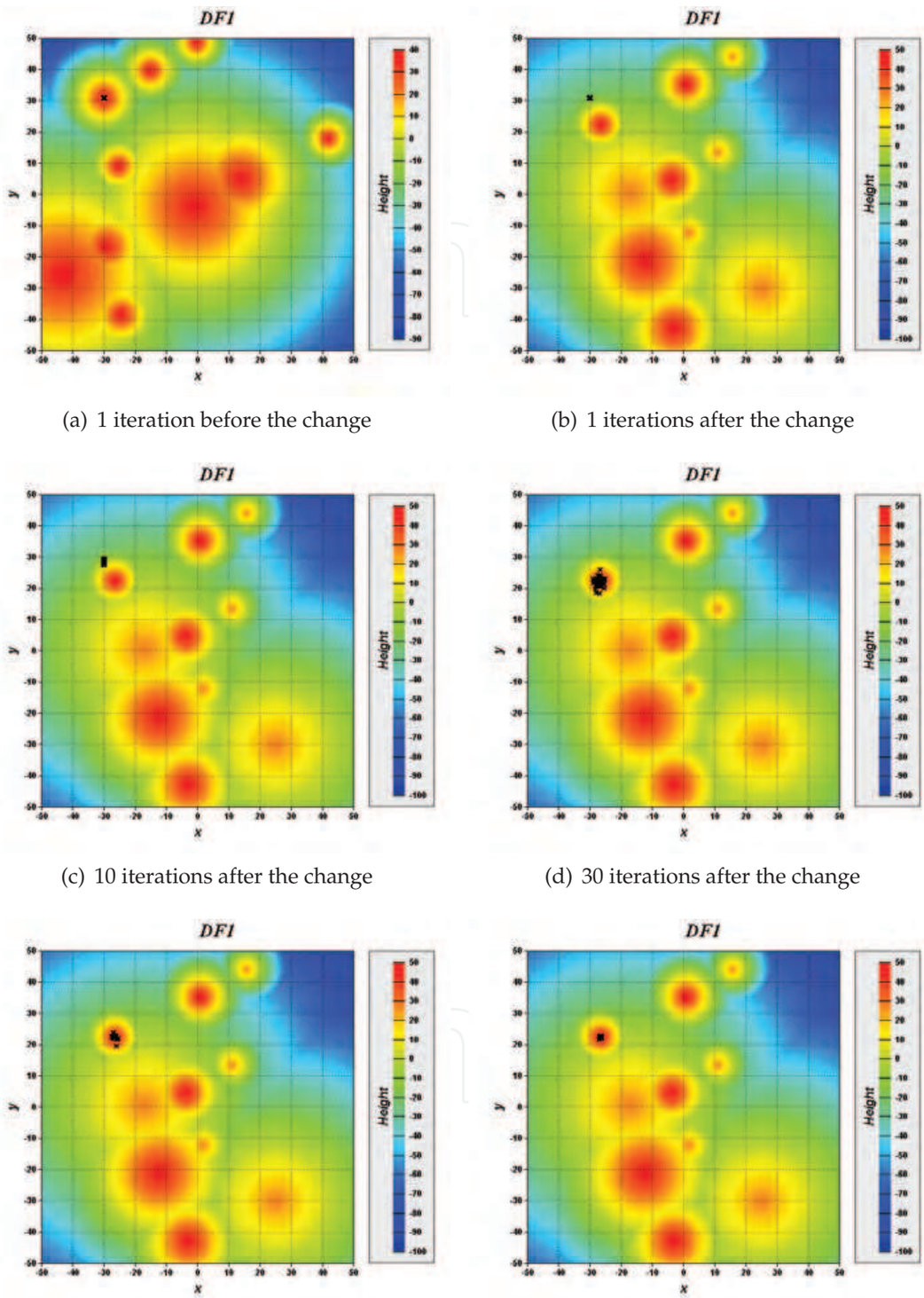
(c) 10 iterations after the change

(d) 30 iterations after the change

(e) 50 iterations after the change

(f) 100 iterations after the change

Fig. 6. Positions of the particles for the *PSO-L* in a dynamic environment.



(a) 1 iteration before the change

(b) 1 iterations after the change

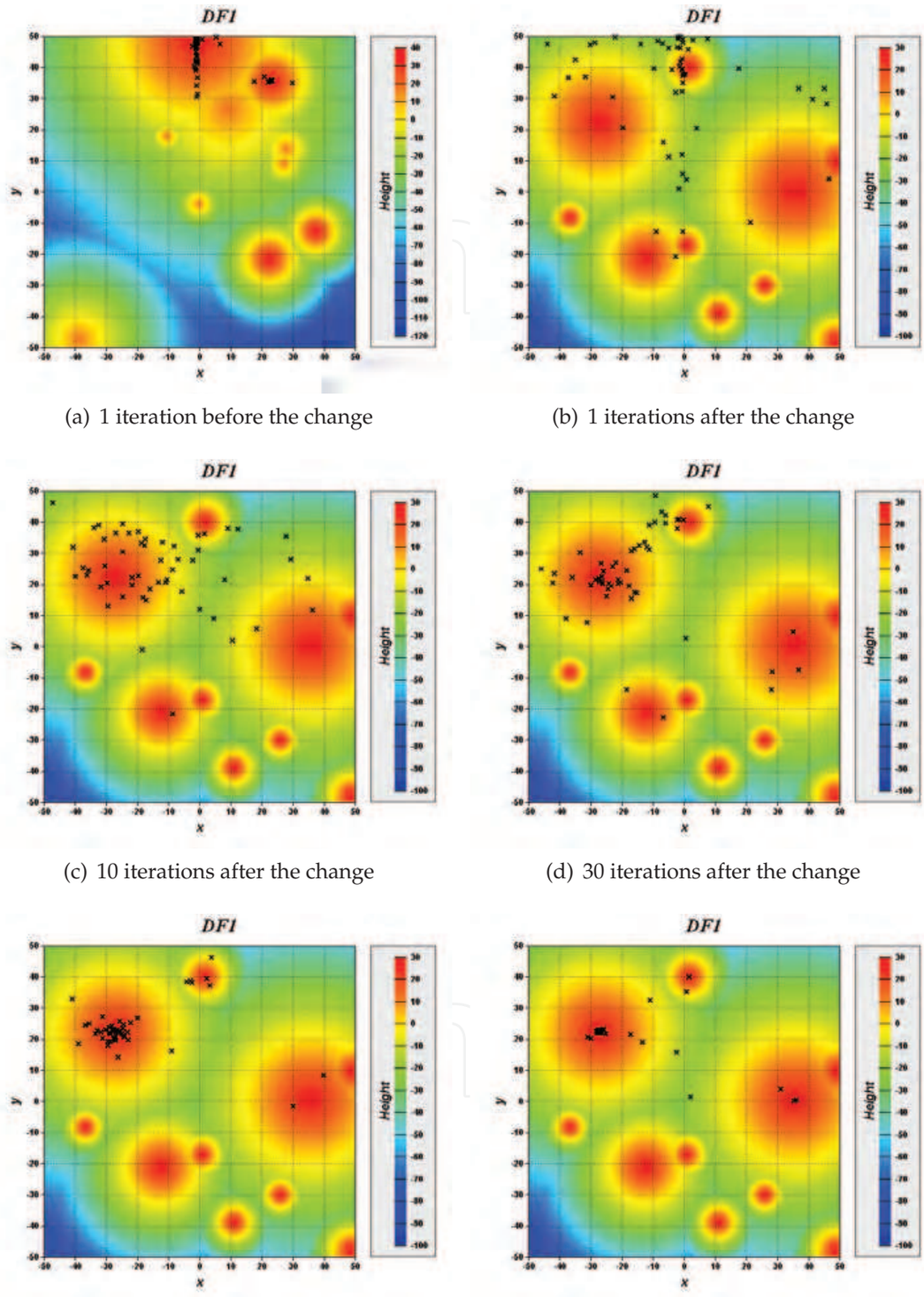
(c) 10 iterations after the change

(d) 30 iterations after the change

(e) 50 iterations after the change

(f) 100 iterations after the change

Fig. 7. Positions of the particles for the *Clan PSO-L 3x18* in a dynamic environment.



(a) 1 iteration before the change

(b) 1 iterations after the change

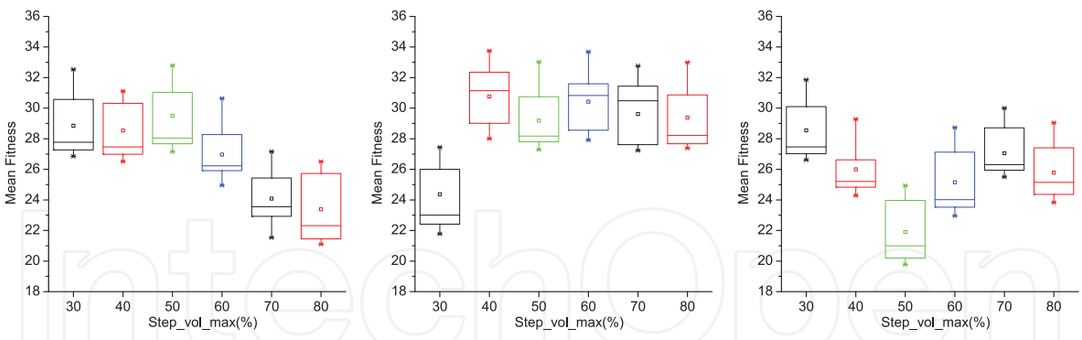
(c) 10 iterations after the change

(d) 30 iterations after the change

(e) 50 iterations after the change

(f) 100 iterations after the change

Fig. 8. Positions of the particles for the *Volitive Clan PSO-L 3x18* in a dynamic environment.



(a) $decay_{vol} = 0\%$ (b) $decay_{vol} = 5\%$ (c) $decay_{vol} = 10\%$

Fig. 9. Box plot of *Volitive Clan PSO 3x18* in the last iteration in high severity environment.

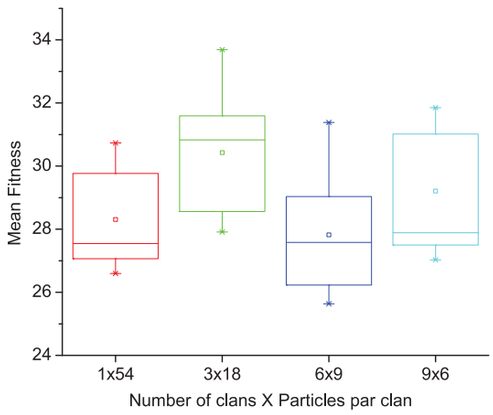
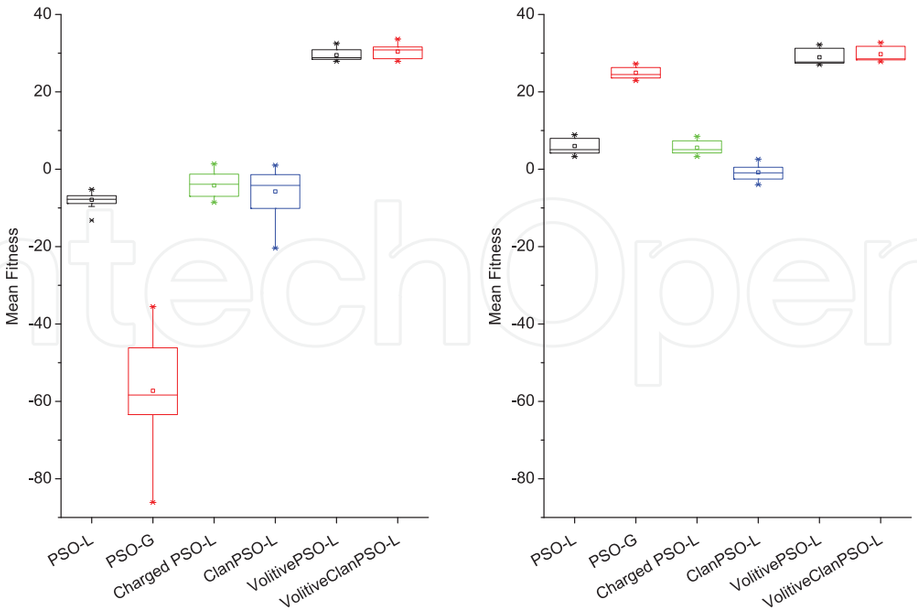


Fig. 10. Performance comparison between different number of clans and particles per clan with $decay_{vol} = 5\%$ and $step_{vol_max} = 60\%$.

reinitialization process slightly mitigated the overall performance. It probably occurs because the mutation causes information loss. Furthermore, it indicates that the volitive operator can generate enough diversity for the presented case. The *Volitive Clan PSO* achieved slightly better results when compared to the *Volitive PSO*. The *PSO-G* improves its performance significantly when using the reinitialization, but the results were worse than the *Volitive PSO* and the *Volitive Clan PSO*.

Figure 12 and Tables 1(a) and 1(b) present the results for the experiments with 10 dimensions and 50 peaks. The results are similar to the ones with 10 dimensions and 10 peaks. Again, the *PSO-G* improves its performance with the reinitialization and achieved results similar to the *Volitive PSO* and the *Volitive Clan PSO*. Nevertheless, the *PSO-G* was dependent on the reinitialization to generate diversity, in this case all reinitialized particles loose their memories (*i.e.* the $\vec{P}_{best_i}$). On the other hand, the *Volitive PSO* and the *Volitive Clan PSO* are not dependent on the reinitialization. Thus, in these algorithms the information acquired previously is not totally lost.

The Table 3 shows the processing time in seconds running 10.000 iterations. The *Charged PSO* reached the smallest processing time. Nevertheless, the processing time among all algorithms are not so different.



(a) Without reinitialize particles after an environment change

(b) Reinitializing 50% of particles after an environment change

Fig. 11. Mean Fitness comparison between five algorithms in environment with 10 dimensions and 10 peaks.

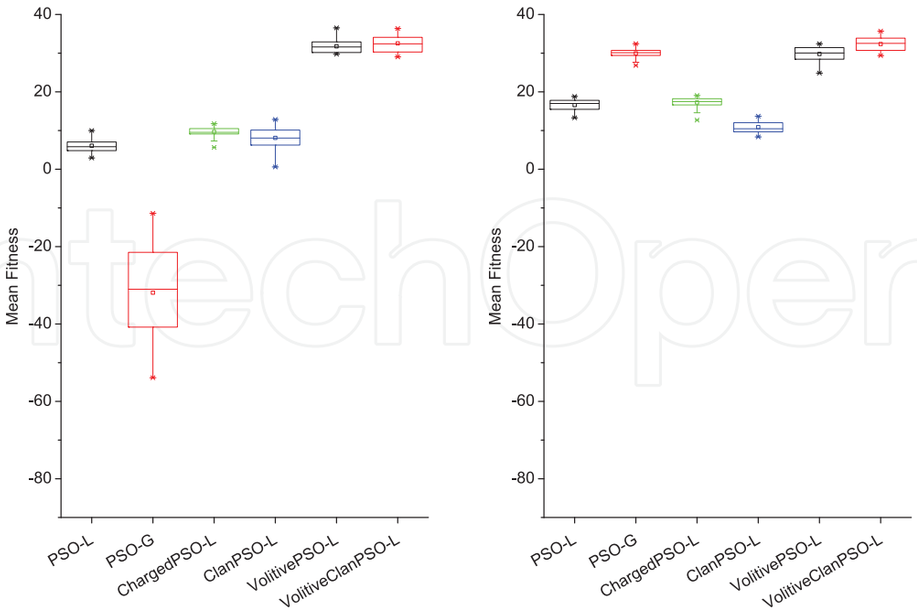
(a) High severity.

Algorithm	Mean	Standard deviation
PSO-L	-7.886	1.589
PSO-G	-57.240	12.699
ChargedPSO-L	-4.175	2.856
ClanPSO-L	-5.740	5.437
VolitivePSO-L	29.496	1.453
VolitiveClanPSO-L	30.430	1.732

(b) High severity with initialization of 50% of particles.

Algorithm	Mean	Standard deviation
PSO-L	6.001	2.022
PSO-G	24.924	1.484
ChargedPSO-L	5.573	1.672
ClanPSO-L	-0.813	1.816
VolitivePSO-L	28.953	2.009
VolitiveClanPSO-L	29.762	1.854

Table 1. Mean fitness in the last iteration with 10 dimensions and 10 peaks - mean and standard deviation after 10,000 iterations.



(a) Without reinitialize particles after an environment change

(b) Reinitializing 50% of particles after an environment change

Fig. 12. Mean Fitness comparison between five algorithms in environment with 10 dimensions and 50 peaks.

(a) High severity.

Algorithm	Mean	Standard deviation
PSO-L	6.082	1.693
PSO-G	-31.905	11.588
ChargedPSO-L	9.658	1.375
ClanPSO-L	8.098	3.210
VolitivePSO-L	31.769	1.682
VolitiveClanPSO-L	32.545	2.119

(b) High severity with initialization of 50% of particles.

Algorithm	Mean	Standard deviation
PSO-L	16.587	1.532
PSO-G	29.947	1.3557
ChargedPSO-L	17.261	1.416
ClanPSO-L	10.865	1.601
VolitivePSO-L	29.807	2.163
VolitiveClanPSO-L	32.372	1.958

Table 2. Mean fitness in the last iteration with 10 dimensions and 50 peaks - mean and standard deviation after 10,000 iterations.

Algorithm	Mean	Standard deviation
PSO-L	11.323 s	0.934
PSO-G	8.545 s	0.253
ChargedPSO-L	7.570 s	0.673
ClanPSO-L	9.433 s	0.387
VolitivePSO-L	11.613 s	0.218
VolitiveClanPSO-L	10.022 s	0.244

Table 3. Processing time of the algorithms in 10 dimensions and 10 peaks - mean and standard deviation after 10,000 iterations.

10. Conclusions

In this chapter we presented a new *PSO*-based approach capable to handle dynamic problems. We achieved this by incorporating the volitive operator in the *Clan PSO*. Our approach is capable to generate diversity without use particles reinitialization. Thus, it does not totally loose information about the environment whenever a change occurs. Actually, the reinitialization of the particles was detrimental for our approach.

We believe that the fast convergence of the *PSO* and the ability of the volitive operator to self-regulate the granularity of the search were responsibly for the success in dealing with dynamic problems. The volitive operator contributes either for diversity and convergence by expanding or shrinking the swarm, then this is another feature that improved the performance of either *PSO* and *Clan PSO*. For all experiments, the *Volitive Clan PSO* outperforms *PSO*, *Clan PSO*, *Charged PSO* and slighly outperforms the *Volitive PSO*.

11. References

Bastos Filho, C. J. a., de Lima Neto, F. B., Lins, A. J. C. C., Nascimento, A. I. S. & Lima, M. P. (2009). A novel search algorithm based on fish school behavior, *2008 IEEE International Conference on Systems, Man and Cybernetics*, IEEE, pp. 2646–2651.

Bastos-Filho, C. J. A., Lima-Neto, F. B., Sousa, M. F. C. & Pontes, M. R. (2009). On the Influence of the Swimming Operators in the Fish School Search Algorithm, *SMC* pp. 5012–5017.

Blackwell, T. & Bentley, P. (2002). Dynamic Search with Charged Swarms, *Proceedings of the Genetic and Evolutionary Computation Conference* pp. 19–26.

Carlisle, A. & Dozier, G. (2002). *Applying the particle swarm optimizer to non-stationary environments*, Phd thesis, Auburn University, Auburn, AL.

Carvalho, D. F. & Bastos-Filho, C. J. A. (2009a). Clan particle swarm optimization, *International Journal of Intelligent Computing and Cybernetics* 2: 197–227.

Carvalho, D. F. D. E. & Bastos-Filho, C. J. A. (2009b). Clan Particle Swarm Optimization, *International Journal Of Intelligent Computing and Cybernetics* pp. 1–35.

Cavalcanti-Júnior, G. M., Bastos-Filho, C. J. A., Lima-Neto, F. B. & Castro, R. M. C. S. (2011). A hybrid algorithm based on fish school search and particle swarm optimzation for dynamic problems, *in* Y. Tan (ed.), *Proceedings of the International Conference on Swarm intelligence*, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Heidelberg, pp. 543–552.

Leonard, B. J., Engelbrecht, A. P. & van Wyk, A. B. (2011). Heterogeneous Particle Swarms in Dynamic Environments, *IEEE Symposium on Swarm Intelligence - SIS, IEEE Symposium Series on Computational Intelligence - SSCI* pp. 9–16.

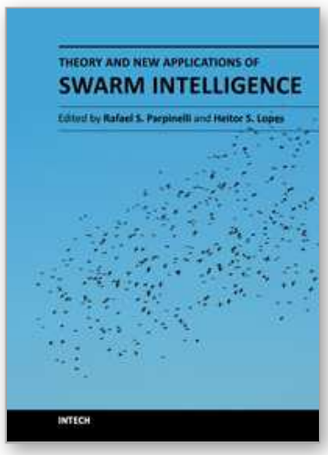
Morrison, R. & De Jong, K. (1999). A test problem generator for non-stationary environments, *Proceedings of the Congress on Evolutionary Computation* pp. 2047–2053.

Morrison, R. W. (2003). Performance Measurement in Dynamic Environments, *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems* pp. 5–8.

Shi, Y. & Eberhart, R. (1998). A modified particle swarm optimizer, *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on* pp. 69–73.

IntechOpen

IntechOpen



Theory and New Applications of Swarm Intelligence

Edited by Dr. Rafael Parpinelli

ISBN 978-953-51-0364-6
Hard cover, 194 pages
Publisher InTech
Published online 16, March, 2012
Published in print edition March, 2012

The field of research that studies the emergent collective intelligence of self-organized and decentralized simple agents is referred to as Swarm Intelligence. It is based on social behavior that can be observed in nature, such as flocks of birds, fish schools and bee hives, where a number of individuals with limited capabilities are able to come to intelligent solutions for complex problems. The computer science community have already learned about the importance of emergent behaviors for complex problem solving. Hence, this book presents some recent advances on Swarm Intelligence, specially on new swarm-based optimization methods and hybrid algorithms for several applications. The content of this book allows the reader to know more both theoretical and technical aspects and applications of Swarm Intelligence.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

George M. Cavalcanti-Júnior, Carmelo J. A. Bastos-Filho and Fernando B. de Lima-Neto (2012). Volitive Clan PSO - An Approach for Dynamic Optimization Combining Particle Swarm Optimization and Fish School Search, Theory and New Applications of Swarm Intelligence, Dr. Rafael Parpinelli (Ed.), ISBN: 978-953-51-0364-6, InTech, Available from: http://www.intechopen.com/books/theory-and-new-applications-of-swarm-intelligence/volitive-clan-pso-an-approach-for-dynamic-optimization-combining-particle-swarm-optimization-and-fis

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2012 The Author(s). Licensee IntechOpen. This is an open access article distributed under the terms of the Creative Commons Attribution 3.0 License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

IntechOpen

IntechOpen