2009 Ninth International Conference on Intelligent Systems Design and Applications

Dynamic Clan Particle Swarm Optimization

C. J. A. Bastos-Filho, D. F. Carvalho, E. M. N. Figueiredo and P. B. C. de Miranda
Department of Computing and Systems, University of Pernambuco, Brazil
cjabf@dsc.upe.br, dfc@dsc.upe.br, emnf@dsc.upe.br, pbcm@dsc.upe.br

Abstract

Particle Swarm Optimization (PSO) has been widely used to solve many different real world optimization problems. Many novel PSO approaches have been proposed to improve the PSO performance. Recently, a communication topology based on Clans was proposed. In this paper, we propose the Dynamic Clan PSO topology. In this approach, a novel ability is included in the Clan Topology, named migration process. The goal is to improve the PSO degree of convergence focusing on the distribution of the particles in the search space. A comparison with the Original Clan topology and other well known topologies was performed and our results in five benchmark functions have shown that the changes can provide better results, except for the Rastrigin function.

1 Introduction

Particle Swarm Optimization (PSO) is a class of bio-inspired algorithms that can be used to solve optimization and search problems. In general, PSO is used to tackle with problems in hyperdimensional spaces where the variables are continuous. PSO was proposed by Kennedy and Eberhart in 1995, inspired by the social behavior of flocks of birds [5].

Each particle i represents a possible solution to a problem and has three main attributes: the position in the search space $\vec{x_i}(t)$, the current velocity $\vec{v_i}(t)$ and the best position ever found by the particle during the search process $\vec{p_i}(t)$ so far. Besides, the particles change information by using a specific communication scheme. The information flows depending on a predefined neighborhood for each particle. The neighborhood defines the communication topology.

The particles move through the search space performing movements based on the best solution already found individually, and the best solution found by the neighborhood which they belong $\vec{p_{ig}}(t)$.

During each algorithm iteration, the particle´s velocities are updated according to the velocity equation. There are, at least, three different equation to update the velocity. The one used in this paper was developed by Clerc [3] and involves the constriction factor presented in the equation below:

$$\begin{aligned}\vec{v_i}(t+1) &= \chi[\vec{v_i}(t) + c_1\epsilon_1(\vec{p_i}(t) - \vec{x_i}(t)) \\ &+ c_2\epsilon_2(\vec{p_{ig}}(t) - \vec{x_i}(t))] \quad , \end{aligned} \tag{1}$$

where

$$\chi = \frac{2\kappa}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}, \varphi = c_1\epsilon_1 + c_2\epsilon_2 \quad , \tag{2}$$

where c_1 and c_2 are positive constants, ϵ_1 and ϵ_2 are two random numbers generated by an uniform distribution in the interval [0,1]. The constriction factor χ assumes a value between 0 and 1, which implies a velocity reduction at each time step. Under the conditions $\varphi \geq 4$ and $\chi \in [0, 1]$, the constriction approach can avoid an unstable state. The parameter κ, in equation (2), controls the exploration-exploitation tradeoff of the swarm. $\kappa \approx 1$ values are used in order to provide a high degree of exploration but with a slow convergence. Lower values of κ are used for faster convergence. However, it leads to a higher degree of exploitation.

The first term in the right side of equation 1 takes into account the current velocity taken by the particle and prevents drastic changes of directions. The second term is the cognitive component and the last one is called the social component.

Another important issue that deserves attention, is the communication topology used to spread information inside the swarm. Many topologies have been proposed and improved to accelerate the convergence process with accuracy. The most common topologies are shown in Fig. 1.

In the *star* topology (Fig. 1(a)), particles can share information globally through a fully-connected structure. This topology uses a global neighborhood mechanism known as *gbest* to share information.

On the other hand, there are topologies based on a local neighborhood, such as Fig. 1(b). This topology is called

978-0-7695-3872-3/09 $26.00 © 2009 IEEE
DOI 10.1109/ISDA.2009.10

ring or *lbest*. In this approach, the particles only share information with their direct neighbors defined based on indexs. One should note thar the neighborhood is not based on the spatial information. The ring topology provides better solutions for multi-modal problems than the star topology [4]. Despite of this, the ring topology needs more iterations to converge.

Some topologies have been proposed to balance the extreme behavior of the *gbest* and *lbest* approaches, such as the Von Neumann topology [6, 8] and clusters [7]. In the Von Neumann topology (presented in Fig. 1(c)), particles are connected by a grid creating a social structure very useful for many optimization problems.

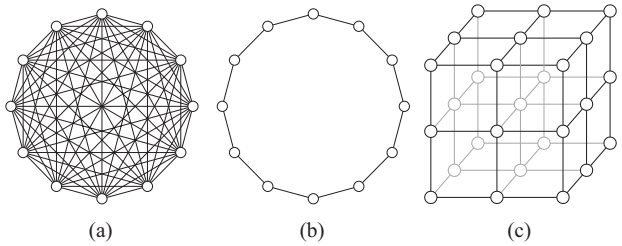

Figure 1. Standard structures: (a) Star topology used in *gbest*, (b) Ring topology used in *lbest*. and (c) Von Neumann Topology.

PSO has been widely used in real world applications due to its simplicity and capacity to deal with optimization problems. However, in some cases the swarm gets trapped in local minima. To avoid this problem, dynamic topologies forming peculiar types of particle grouping, such as rings [1] and clans [2], have been considered to dynamically change the information flows inside the swarm in order to increase the diversity when a stagnation process occurs.

This paper aims to include some extra abilities to the Clan Particle Swarm Optimization. The paper is organized as follow. In section 2, we present the basic concept of Clan PSO. In section 3 we present our contributions in order to allow migration processes among the clans. In section 4 and 5 we present the simulation setup and the results, respectively. In section 6 we give our conclusions.

2 Clan Particle Swarm Optimization

This section reviews briefly a recently proposed topology called Clan Particle Swarm Optimization (Clan PSO) [2]. These concepts will be necessary to explain the novel abilities included in this approach.

Clan PSO was inspired by social behaviour of Clans and was developed to improve the PSO performance. The clans are formed by groups of particles in a fully-connected structure to share information globally (*gbest*). The structure presented in Fig. 2 is an example with four Clans (A, B, C and D) of 5 particles.

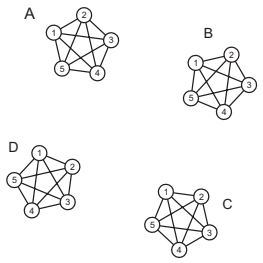

Figure 2. Clan topology: distribution of clans.

In each iteration, each clan performs a search and marks the particle with the current best position $\overrightarrow{p_i}(t)$ of the entire clan. These particles are the current Leaders. The delegation process uses the *gbest* information to directly delegate the leader. Therefore, no additional processing is necessary to perform the delegation.

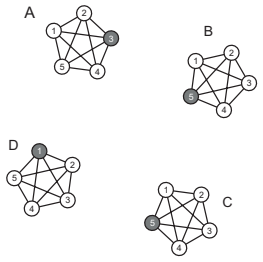

Figure 3. Clan topology: marked leaders of each clan.

After that, the leaders of each clan are put together in a "conference", and another PSO solely with the leaders is ran. The Leader's Conference is performed using a PSO and can use either *gbest* or *lbest* information sharing mechanism as presented in Fig. 4. After the conference, the new information acquired by the leaders in the conference will be used inside each clan to adjust the velocities of the other particles. One should note that the leaders do not acquire the best position found by the other leaders so far. Indeed, the leaders solely adjust their positions based on the best position found by the other leaders.

3 Dynamic Clan Particle Swarm Optimization

In the original Clan Particle Swarm Optimization, it is necessary to define the number of particles in each Clan. Each set of particles that compose a Clan will remain in the

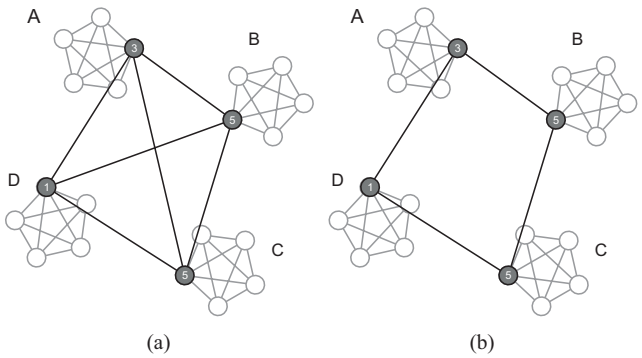

Figure 4. Leader's conference: (a) global conference and (b) local conference.

same Clan until the end of the algorithm execution. Furthermore, the number of particles in each Clan impacts on the performance of the algorithm and the optimum number of particles per Clan depends on the problem.

Considering that some particles are not necessarily spatially close to its own Clan, it would be interesting for these particles to be influenced by other Clans. Besides, as the particles will be free to move out and join different Clans, under certain conditions, the number of particles in a determined Clan can increase, leading to a high degree of exploitation in the region covered by the Clan.

Because of these reasons, we are proposing to create a mechanism to allow particles to migrate from one Clan to another Clan during the search process. We call this new ability *Migration Process*, which was included into the Clan PSO algorithm. The Dynamic Clan PSO pseudo-code is presented in Algorithm 1.

Algorithm 1: Dynamic Clan algorithm search process pseudo-code.

```
1  Initialize Swarm
2  while step <= number Of Steps do
3      foreach clan of the swarm do
4          foreach particle of the clan do
5              Update Velocity And Position
6              Calculate Fitness
7              Update Information
8          end
9          Delegate Clan Leader
10     end
11     Perform Leaders' Conference
12     Perform Particles' Migration
13 end
14 return Best Position Found
```

3.1 Migration Process

Consider that the particle P belongs to a Clan C_i in the search space containing n clans. During the migration process, the euclidian distance in the search space $d(P, L_i)$ between particle P and the leader L_i of its own clan (C_i) will be computed. If $d(P, L_i) > d(P, L_j)$ where $j \in \{0, 1, \ldots, n\}$ and $i \neq j$, P will migrate to the clan whose leader is spatially closer, *i.e.* the clan j. It aims to reorganize the Clans with particles which are spatially close.

The migrations just can occur in predefined iterations called *epochs*. This article proposes two different modes of operation. The exploration mode and the exploitation mode. In the exploration mode, the predefined iterations (*exploration epoch*) in which the particles can migrate have higher values, tipically 1,000 iterations. In the exploitation mode, the predefined iterations (*exploitation epoch*) in which the particles can migrate have lower values, tipically 100, 200 or 400 iterations. The *exploration epoch* is used during the most part of the PSO algorithm execution, and the *exploitation epoch* is used mostly in the end of the simulation. Another parameter used in our algorithm is the iteration in which the algorithm switches from exploration mode to exploitation mode.

A problem can arise from the migration process. After some epochs, all particles can be close to each other. In this case, the Dynamic Clan topology tends to behave like a *star* topology. Thus, we define the minimum number of particles per Clan s_{min} to avoid this problem. It means that clans can not lose particles when the s_{min} threshold has been reached. One should note that it can avoid an extreme behaviour.

4 Experiments

To perform the experiments, five benchmark functions were chosen to be used in simulations and many simulations were performed to analyse which values would be better for the parameters s_{min}, *exploration epoch* and the *exploitation epoch*.

4.1 Benchmark Functions

Five benchmark functions were used in the simulations. All five functions are used for minimization problems. Two of these functions, Rosenbrock (f_1) and Schwefel 1.2 (f_5), are simple unimodal problems, and the others, Rastrigin (f_2), Griewank (f_3), and Ackley (f_4), are multimodal functions that contain many local optima. Table 1 shows the search space, initialization range, and the optimum for each function. All the simulations were performed using 30 dimensions for all functions.

Table 1. Used functions, search space, initialization range, and optimum.

Function	Search space	Initialization	Opt.
f_1	$-30 \leq x_i \leq 30$	$15 \leq x_i \leq 30$	1.0^D
f_2	$-5.12 \leq x_i \leq 5.12$	$2.56 \leq x_i \leq 5.12$	0.0^D
f_3	$-300 \leq x_i \leq 600$	$300 \leq x_i \leq 600$	0.0^D
f_4	$-32 \leq x_i \leq 32$	$16 \leq x_i \leq 32$	0.0^D
f_5	$-100 \leq x_i \leq 100$	$50 \leq x_i \leq 100$	0.0^D

4.2 Simulations setup

The equation used to update the particle´s velocities is the constricted PSO. We used $\kappa = 1$ and $c_1 = c_2 = 2.05$. All the simulations were performed using 30 particles and $10,000$ iterations.

The parameters involved in the *migration process* were chosen based on preliminar simulation results. In general, the parameters are: $s_{mim} = 4$, the *exploration epoch* is $1,000$ iterations until the $8,000^{th}$ iteration and the *exploitation epoch* is 200 iterations from the $8,000^{th}$ iteration until the end of the algorithm execution.

All the results presented in next section were run 30 times. In all the cases, the mean value and standard deviation of the fitness values were evaluated after 30 trials.

5 Results

We performed two sets of experiments to determine the best values for the parameters used in the migration process. After that, we performed a comparison between our approach and previous approaches.

5.1 Analysis of the Parameters Influence

The first analysis aims to define the best values for the *exploration epochs*, the *exploitation epochs* and the iteration to switch from exploration mode to exploitation mode. The results are shown in Table 2. The mean value and the standard deviation are presented for five benchmark functions, two types of leaders conference and four different epochs configuration. The second row presents the parameters used to define the frequency of the migration process. These simulations used $s_{min} = 3$.

Considering the cases presented in Table 2, there are no significant variations in the results by changing the *exploration epochs*, the *exploitation epochs* and the iteration to switch from exploration mode to exploitation mode. Thus, we adopted in the rest of this paper *exploration epochs*=$1,000$, the *exploitation epochs*=200 and the =$8,000^{th}$ iteration to switch from exploration mode to exploitation mode.

The second analysis was performed in order to define the best value for s_{min}. Table 3 shows the results for $s_{min} = 2$, 3, 4 and 5. The same explanation used for Table 2, can be applied to Table 3. However, in the column Fitness, the mean and standard deviation were evaluated for different values of s_{min}. Again, we detected that s_{min} is not a critical parameter. Despite of this, we checked that in general the algorithm achieves satisfactory results for $s_{min} = 4$. Therefore, we will use this value in the rest of the paper.

5.2 Comparison to other Topologies

In this subsection we present a comparison between the Dynamic Clan PSO, using the parameters which were defined above (s_{min}, *exploration epoch* and *exploitation epoch*), and other topologies such as Global, Local, Von Neumann and original Clan topology. The results of mean values and standard deviation for the fitness are shown Tables 4, 5, 6, 7 and 8.

As can be seen in Table 4, the Dynamic Clan PSO with Global Leaders conference achieved the best results (*mean* = 2.64 and *standard deviation* = 2.76). The Dynamic Clan PSO with Local Leaders conference obtained results similar to original Clan PSO with Local Leaders conference and Star topology.

Table 4. Topologies comparation for Rosenbrock function.

Simulation		Fitness	
Function	Topology	Mean	S. D.
Rosenbrock	DynClan-Global	2.64	2.76
	DynClan-Local	4.49	4.15
	Clan-Global	7.39	18.20
	Clan-Local	4.54	4.11
	Star	4.50	12.54
	Ring	9.79	7.82
	Von Neumann	6.92	5.24

The results for Rastrigin function were not as good as the results for the Original Clan PSO, as shows the Table 5. Despite of this, both the Dynamic Clan PSO with Global and Local Leaders conference outperformed the other topologies. A possible solution to improve the Dynamic Clan convergence would be to increase the *exploratory epoch*, allowing the particles to explore the search space and escape from local minima.

The results for the Schwefell 1.2 function (see Table 6) show that the Dynamic Clan PSO outperformed the original Clan PSO. Despite the Global topology achieved the best

Table 2. Dynamic-Clan simulation results for epoch values variation.

		Fitness							
Exploration/Exploitation/Switch		1,000/200/8,000		1,000/400/8,000		1,000/100/9,000		1,000/200/9,000	
Function	Topology	Mean	S. D.	Mean	S. D.	Mean	S. D.	Mean	S. D.
Rosenbrock	DynClan-Global	2.49	2.88	2.63	3.44	1.79	2.59	5.05	13.79
	DynClan-Local	7.08	12.40	9.76	19.09	4.20	3.90	5.41	4.14
Rastrigin	DynClan-Global	35.93	12.17	33.75	14.21	35.30	14.78	33.92	11.09
	DynClan-Local	28.75	11.07	30.53	15.81	28.80	12.81	25.61	9.08
Schwefel 1.2	DynClan-Global	7.4E-18	1.9E-17	1.6E-17	4.5E–17	1.0E-18	2.5E-18	1.2E-17	4.2E-17
	DynClan-Local	1.9E-13	3.8E-13	4.4E-13	9.3E-13	1.9E-13	5.5E-13	8.8E-13	3.3E-12
Ackley	DynClan-Global	12.19	8.99	15.64	7.84	18.02	5.73	17.78	7.12
	DynClan-Local	13.48	9.82	11.46	9.25	10.99	9.71	12.77	9.52
Griewank	DynClan-Global	0.02	0.02	0.03	0.06	0.02	0.03	0.01	0.02
	DynClan-Local	0.01	0.03	0.02	0.04	0.01	0.01	0.02	0.02

Table 3. Dynamic-Clan simulation results for s_{min} variation.

		Fitness							
Minimum Number of Particles		2		3		4		5	
Function	Topology	Mean	S. D.	Mean	S. D.	Mean	S. D.	Mean	S. D.
Rosenbrock	DynClan-Global	2.58	2.91	2.49	2.88	2.64	2.76	4.31	12.39
	DynClan-Local	7.91	14.02	7.08	2.40	4.49	4.15	4.43	4.14
Rastrigin	DynClan-Global	44.50	17.19	35.93	12.17	33.97	16.16	23.55	10.65
	DynClan-Local	34.09	12.28	28.75	11.07	24.65	9.24	20.83	7.80
Schwefel 1.2	DynClan-Global	5.3E-18	1.4E-17	7.1E-18	1.9E-17	3.5E-17	1.2E-16	3.6E-17	8.5E-17
	DynClan-Local	1.4E-13	2.5E-13	1.9E-13	3.8E-13	4.7E-12	2.0E-11	1.1E-12	1.4E-12
Ackley	DynClan-Global	11.23	9.44	12.19	8.99	14.91	8.44	13.62	9.16
	DynClan-Local	14.76	8.66	13.48	9.25	12.94	9.31	16.67	7.36
Griewank	DynClan-Global	0.02	0.02	0.02	0.02	0.02	0.02	0.05	0.11
	DynClan-Local	0.01	0.02	0.01	0.03	0.01	0.01	0.01	0.01

Table 5. Topologies comparation for Rastrigin function.

Simulation		Fitness	
Function	Topology	Mean	S. D.
Rastrigin	DynClan-Global	33.97	16.16
	DynClan-Local	24.65	9.24
	Clan-Global	11.84	7.75
	Clan-Local	6.10	3.67
	Star	52.26	14.34
	Ring	37.98	6.58
	Von Neumann	34.75	9.52

Table 6. Topologies comparation for Schwefel 1.2 function.

Simulation		Fitness	
Function	Topology	Mean	S. D.
Schwefel 1.2	DynClan-Global	3.5E-17	1.2E-16
	DynClan-Local	4.7E-12	2.0E-11
	Clan-Global	4.3E-12	2.3E-11
	Clan-Local	4.8E-10	1.1E-9
	Star	1.5E-21	3.6E-21
	Ring	3.0E-4	2.0E-4
	Von Neumann	6.9E-9	1.1E-8

results in this case, the Dynamic Clan PSO overcame Von Neumann and Ring topologies.

The results for the Ackley function (see Table 7) show that, on average, the Dynamic Clan PSO outperformed the other topologies. However, the higher standard deviation values indicate that in some trials the Dynamic Clan PSO got trapped in local minima. The number of times that the Dynamic Clan PSO gets trapped on local minima is lower

than the number of times that the Original Clan PSO gets trapped on local minima. One should note that the other topologies are not able avoid the local minima in all the trials.

Table 7. Topologies comparation for Ackley function.

Simulation		Fitness	
Function	Topology	Mean	S. D.
Ackley	DynClan-Global	14.91	8.44
	DynClan-Local	12.94	9.31
	Clan-Global	17.31	6.66
	Clan-Local	15.99	7.90
	Star	19.91	0.03
	Ring	19.91	0.03
	Von Neumann	19.93	0.01

The results for the Griewank function (see Table 8) show that the Dynamic Clan PSO achieved similar results when compared to the original Clan PSO. Although the best results for this function were achieved by the Von Neumann topology, the Clan based topologies outperformed the Ring and Star topologies.

Table 8. Topologies comparation for Griewank function.

Simulation		Fitness	
Function	Topology	Mean	S. D.
Griewank	DynClan-Global	0.02	0.02
	DynClan-Local	0.008	0.01
	Clan-Global	0.01	0.01
	Clan-Local	0.008	0.01
	Star	0.03	0.04
	Ring	0.77	0.86
	Von Neumann	0.005	0.006

6 Conclusions

In this paper we proposed to include a novel ability to a recently proposed communication topology for Particle Swarm Optimization, the Clan Particle Swarm Optimization. This ability allows the particles of one Clan to migrate to another Clan. We believe that this process can help to automatically increase the exploitation capacity of a subswarm when it is necessary.

This migration process can occur in some moments during the algorithm execution, called epochs. Besides, we divided the search process in two phases, the exploration mode and the exploitation mode. Three parameters controls the migration process scheme: the *exploration epochs*, the *exploitation epochs* and the iteration to switch from exploration mode to exploitation mode. The simulation results indicated that these parameters are not critical in the studied range.

The simulation results also showed that the proposed topology achieved similar or superior performance in almost all the benchmark functions considered in this paper. Furthermore, the migration process improved the performance of the Clan PSO in three cases. The only exception occured for Rastrigin function. However, we believe that further investigations on the relationship between the parameters involved in the migration process can mitigate this problem.

References

[1] C. J. A. Bastos-Filho, M. P. Caraciolo, P. B. C. Miranda, and D. F. Carvalho. Multi-ring particle swarm optimization. In *SBRN '08: Proceedings of the 2008 10th Brazilian Symposium on Neural Networks*, pages 111–116, Washington, DC, USA, 2008. IEEE Computer Society.

[2] D. F. Carvalho and C. J. A. Bastos-Filho. Clan particle swarm optimization. *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 3044–3051, June 2008.

[3] M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE-EC*, 6:58–73, Feb. 2002.

[4] A. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. John Wiley & Sons, 2005.

[5] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proc. of the IEEE Int. Conf. on Neural Networks*, pages 1942–1948, Piscataway, NJ, 1995. IEEE Service Center.

[6] J. Kennedy and R. Mendes. Population structure and particle swarm performance. In *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pages 1671–1676. IEEE Press, 2002.

[7] R. Mendes, J. Kennedy, and J. Neves. Watch thy neighbor or how the swarm can learn from its environment. In *Swarm Intelligence Symposium, 2003. SIS '03. Proceedings of the 2003 IEEE*, pages 88–94, Apr. 2003.

[8] E. Peer, F. van den Bergh, and A. Engelbrecht. Using neighbourhoods with the guaranteed convergence pso. *Swarm Intelligence Symposium, 2003. SIS '03. Proceedings of the 2003 IEEE*, pages 235–242, 24-26 April 2003.