

Adaptative Clustering Particle Swarm Optimization

Salomão S. Madeiro, Carmelo J. A. Bastos-Filho, *Member, IEEE*, and Fernando B. Lima Neto, Senior Member, IEEE, Elliackin M. N. Figueiredo

Abstract—The performance of Particle Swarm Optimization (PSO) algorithms depends strongly upon the interaction among the particles. The existing communication topologies for PSO (e.g. star, ring, wheel, pyramid, von Neumann, clan, four clusters) can be viewed as distinct means to coordinate the information flow within the swarm. Overall, each particle exerts some influence among others placed in its immediate neighborhood or even in different neighborhoods, depending on the communication schema (rules) used. The neighborhood of particles within PSO topologies is determined by the particles' indexes that usually reflect a spatial arrangement. In this paper, in addition to position information of particles, we investigate the use of adaptive density-based clustering algorithm – ADACLUS – to create neighborhoods (i.e. clusters) that are formed considering velocity information of particles. Additionally, we suggest that the new clustering rationale be used in conjunction with Clan-PSO main ideas. The proposed approach was tested in a wide range of well known benchmark functions. The experimental results obtained indicate that this new approach can improve the global search ability of the PSO technique.

I. INTRODUCTION

PARTICLE Swarm Optimization is a population-based technique of Computational Intelligence proposed by Kennedy and Eberhart in 1995 [1]. The PSO performance is mainly influenced by the interaction among the individuals (particles) of the population (swarm). The hypothesis to create PSO assumed that the information exchanged among the particles on the search space can help to guide other particles towards best regions of the search space, along the time.

The higher is the degree of connectivity among particles, the faster is the convergence of the swarm to regions surrounding possible good solutions. This occurs because the swarm is quickly allowed to exploits specific regions of the search space. However, this fast convergence may also increase the probability of the swarm to be trapped in local minima. As opposed to that, slower convergence can allow the swarm to explore the search space more carefully, leading the swarm, in average, to avoid local optimal and find the best regions. Therefore, it is important to set adequately the degree of connectivity among particles in order to obtain a good balance between exploration and exploitation abilities out of the algorithm.

The particles' neighborhood defines the influence of a

particle on others, determining the PSO topology. Each topology represents different ways to establish the information flow throughout the swarm. In other words, when a particle finds out a good region in the search space, the topology defines how this information will be spread out among the suitable particles. For instance, the original PSO proposed by Kennedy and Eberhart [1] uses the star topology. This means that information about good regions of the search space is broadcasted from a particle to all others instantaneously.

The neighborhood is usually determined based on the spatial location of the particles. Some previous approaches tried to use the Euclidian distance between the particles in order to establish their neighborhood. Suganthan [2] proposed to replace the global best solution by a local best one according to a dynamically neighborhood that increases over iterations. By gradually increasing the neighborhood size based on the distances between the particles, the newly formed neighborhoods tend to include all particles during the final stages of the search process. This causes the local best solution to become the global one [2].

By defining the particles' neighborhood based on their position in the search space, one can conceive a neighborhood for a particle by considering other particles that are moving to different regions of the search space. That is, particles that are spatially close – but only temporarily. In contrast to that, by considering in the same cluster only particles that are moving to similar regions on the search space seems to be much more appropriate than defining the neighborhood based simplistically on the particles' position.

In this paper, we analyze the use of an adaptive density-based clustering algorithm, named ADACLUS (ADaptive CLUstering) [10], to the task of neighborhood formation in PSO. The ADACLUS algorithm is applied to the swarm in order to identify clusters based on the velocity or spatial position of the particles. Each cluster identified by ADACLUS algorithm corresponds to a neighborhood of particles in our approach. At last, each created neighborhood of particles corresponds to a new subswarm and will perform an independent search process.

In order to balance the search modes (i.e. exploration and exploitation), we use a similar approach as the one proposed in Clan Particle Swarm Optimization [3]. In that approach, the particles are grouped in clans and each clan performs a local and independent search process. After each algorithmic iteration, the *gbest* of each clan is considered part of a new swarm (i.e. “leaders' convention”), in which a new iteration of the algorithm is performed using the star or ring topology.

This work was supported by FACEPE and CNPq.

S. S. Madeiro, C. J. A. Bastos-Filho, F. B. Lima Neto and E. M. N. Figueiredo are with the Department of Computing and Systems, University of Pernambuco, Recife-Brazil (e-mail: ssm@dsc.upe.br, cjabf@dsc.upe.br, fbln@dsc.upe.br, emnf@dsc.upe.br).

As for now, we predefined the number of iterations for which the local and the conference search processes are to be carried out.

To assess the joint potential of all ideas put together here, we have carried out a number of simulations on a set of well known benchmark functions. Next, we compared the achieved results with those obtained by some variations of PSO. As will be discussed in Section VI, our approach can improve the global search ability of the PSO.

The other sections of this paper are organized as follows: in Section II, we describe the particle swarm optimizers that are used in our approach. In Section III, we explain some classical topologies for PSO and the one that inspired us most to propose our approach, i.e. Clan-PSO. In Section IV, we comment on the ADACLUS algorithm. In Section V, we detail our ideas. The results of the all experiments are presented in Section VI; Section VII sums up this paper.

II. PARTICLE SWARM OPTIMIZERS

A. Standard PSO

Swarm intelligence is a fairly new area in computational intelligence whose techniques are inspired in the emergent collective and collaborative behavior of social animals such as ants, birds and fish schools; representative algorithms of these meta-heuristics in respective order are: Ant Colony Optimization [4], Particle Swarm Optimization [1] and Fish School Search [12].

PSO algorithm was originally proposed by Kennedy and Eberhart [1] and is largely applied to determine optimal solutions for non-linear functions in continuous search spaces. An additional parameter ω was later introduced by Shi and Eberhart [5] in order to better balance exploration and exploitation abilities within the technique. In PSO, the swarm of particles represents a set of potential solutions for an optimization problem and each particle's behavior is separately governed by two adaptive physical properties, velocity and position, according to (1) and (2), respectively. Hence, the particles "fly" throughout the function domain, guided by these two equations, towards best solutions along the simulation cycles.

$$\vec{v}_{t+1} = \omega \vec{v}_t + c_1 r_1 (\vec{P}_{il} - \vec{x}_t) + c_2 r_2 (\vec{P}_{ig} - \vec{x}_t) \quad (1)$$

$$\vec{x}_{t+1} = \vec{x}_t + \vec{v}_{t+1}, \quad (2)$$

where r_1 and r_2 are random values computed for each dimension and uniformly distributed in $[0, \dots, 1]$. They are used to promote disturbance on the swarm and to generate diversity, which is important to avoid premature convergence. In (1), \vec{P}_{il} is a vector that represents the best position found by each particle during the search process, whereas \vec{P}_{ig} is the best solution found by particle i in its neighborhood. ω is the inertial factor and represents a direct means to influence the convergence time – that is the greater

ω value is, the more t iterations will be necessary for the PSO to achieve convergence-. c_1 and c_2 are the cognitive and the social parameters respectively. Together they regulate the overall functioning of the algorithm. In general they need to be adjusted to every specific optimization problems. For more information about parameter selection and convergence processes of PSO see [6].

B. Constricted PSO

Clerc and Kennedy [7] analyzed the particle's trajectory according to an algebraic perspective as well as in an analytical perspective. After these analyses, they suggested a set of coefficients to balance and control the convergence of the PSO algorithm. From their work, Equations (3) and (4) were derived and given as follows.

$$\vec{v}_i = \chi (\vec{v}_i + c_1 r_1 (\vec{P}_i - \vec{x}_i) + c_2 r_2 (\vec{P}_g - \vec{x}_i)) \quad (3)$$

$$\chi = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}, \varphi = c_1 + c_2, \quad (4)$$

where χ is known as the constriction coefficient. It is deemed to be important, among other reasons, because it controls the growth of each particle's velocity and, depending on the values of φ , it guarantees the convergence of the swarm. Clerc and Kennedy [7] investigated the behavior of the swarm for different values of φ and demonstrated that for $\varphi < 4$ the swarm would move in spiral towards and around \vec{P}_g without any guaranteed convergence. Conversely, if $\varphi > 4$, convergence to \vec{P}_g would be guaranteed and fast.

Eberhart and Shi [8] have shown that a generalized model of PSO with a constriction coefficient is a special case of the PSO algorithm with inertia weight for which the values of the coefficients have been determined analytically. In summary, Clerc and Kennedy [7] suggested that these coefficients do in fact increase the ability of PSO of locating optimal points.

C. Guaranteed Convergence PSO

For the previous Particle Swarm optimizers presented here, each time a particle locates a point better than \vec{P}_g (i.e. $\vec{x}_i = \vec{P}_i = \vec{P}_g$), the velocity update equation is dependant on the value of the velocity on the previous iteration. This means that, when a particle approaches the potential global best solution, the value of its velocity can decay exponentially, leading the swarm to premature converge into a possibly local optima.

In order for keeping the global best particle on moving towards possibly global best solution, Van den Bergh and Engelbrecht proposed a modified Particle Swarm optimizer named Guaranteed Convergence PSO (GCPSO) [9]. For the GCPSO, the velocity update equation is redefined only for the global best particle τ , given by (5).

$$\vec{v}_\tau = -\vec{x}_\tau + \vec{P}_i + \omega \vec{v}_\tau + \rho_\tau (1 - 2r_2) \quad (5)$$

$$\rho_\tau = \begin{cases} 2\rho & \text{if \#successes} > s_c \\ 0.5\rho & \text{if \#failures} > f_c \\ \rho & \text{otherwise} \end{cases} \quad (6)$$

According to that new velocity update equation, the new equation for the global best particle is given by (7).

$$\vec{x}_\tau = \vec{x}_\tau + \vec{v}_\tau \Rightarrow \vec{x}_\tau = \vec{P}_i + \omega \vec{v}_\tau + \rho (1 - 2r_2) \quad (7)$$

In (6), $\#successes$ and $\#failures$ correspond to the number of consecutive successes or failures, respectively. So that, when success occurs, the number of failures must be set to zero. On the contrary, when failure occurs, the number of successes must be zero. For the initial value of ρ , Van den Bergh and Engelbrecht suggested $\rho = 1.0$ and the recommended $s_c = 15$ and $f_c = 5$, respectively [9].

The GCPSO can be especially useful when the number of particles in a swarm is relatively small (e.g. one or two particles), once the velocity of the entire swarm would tend to zero exponentially.

III. SOCIAL STRUCTURES AND INFORMATION FLOW IN PSO

A. Classical Social Structures in PSO

In this section, we highlight some details of the well-known PSO social structures: star, ring, and four clusters.

For the ring network structure, each particle can update its position based on the best solution found by its n_i immediate neighbors. Fig. 1(A) shows the case in which $n_i = 2$. The overlapping of neighborhoods is the means that information about good solutions, found by the swarm, flows onto the particles. The lower the value of n_i , the slower the convergence of the swarm, however larger portions of the search domain are covered when compared to greater values of n_i . This simple social structure produces better solutions for multi-modal problems, for which the swarm needs to avoid local optimal solutions.

The star topology in PSO is a particular case of the ring topology in which $n_i = N - 1$, where N is the population size. The star network structure presents the highest degree of connectivity among the particles. For that topology, all particles know each other's positions and can adjust their own positions based on the best solution found so far by the entire swarm. Therefore, all the particles in the swarm tend to follow a unique leader (*i.e.* the global best solution). Hence, if the global best solution does not change for a number of iterations, the entire swarm can converge around that position in a relatively small number of iterations. The star network structure is more appropriate for unimodal optimization problems. Fig. 1(B) illustrates the star

topology.

As for the four clusters social structure in PSO, the created clusters are connected as illustrated in Fig. 1(C). Each cluster communicates with each other through only one particle. This particle can be viewed as an informer between clusters. Thus, the information about better solutions found by a particular cluster flows to the other clusters through only one informant. If the information transmitted from one informant of cluster C_1 to the other informant of cluster C_2 is better than that in cluster C_2 , the particles in C_2 would adjust their positions according to that new information. The selected particles of each cluster communicate always with the same particle of the other clusters. On the other hand, the particles within a cluster are fully connected to all other particles of the same cluster. So, in the four clusters social structure, each cluster performs exploitation whereas the network structure as a whole is the responsible for exploration.

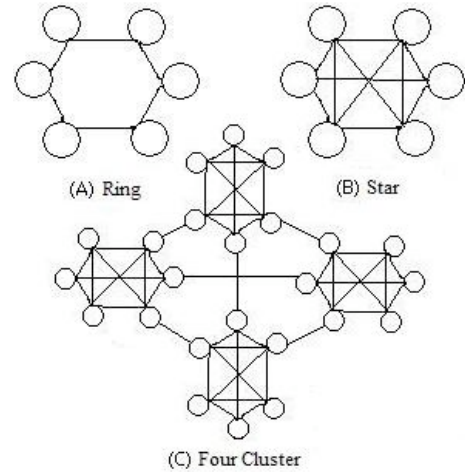


Fig. 1. PSO topologies: (A) ring, (B) star and (C) four clusters.

B. Clan Social Structure in PSO

Clan topology is a novel topology proposed by Carvalho and Bastos-Filho [3] that improves the PSO algorithm performance. It is inspired in the social organization of clans or tribes. Clan topology includes groups of individuals with strong leadership characteristics.

Each clan is formed by a group of particles that use the star topology (*i.e.* fully connected) as illustrated in Fig. 2(A). After each iteration, every clan elects the best solution among all particles of the clan. Each elected particle is selected to be a part of a virtual new smaller swarm (*i.e.* PSO) due to globally search the space using only the current leaders of all clans, as shown in Fig. 2(B). The main idea is that the leaders should adjust their positions according to the position of the best leader to be found in a sort of conference of leaders.

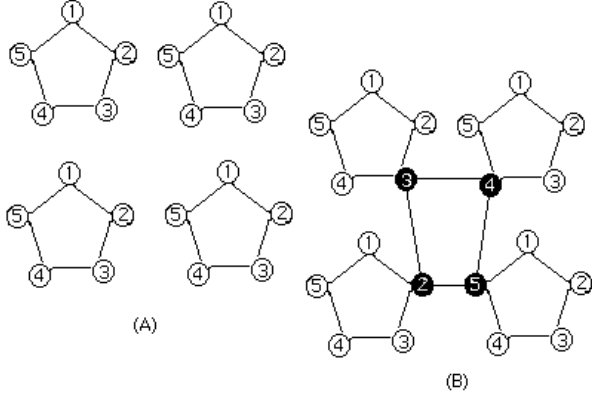


Fig. 2. Clan-PSO topology. (A) Clan search and (B) leaders conference.

After the iteration in which the leaders update their positions, they “return” to their respective clans, where they convey the newly acquired information. The global search results are then distributed locally to the other members of the clan.

IV. ADAPTIVE CLUSTERING

Clustering has been largely applied in computer graphics, pattern recognition, databases, and many other areas of computing. In general, each new application brings some new challenge such as automatic discovery of clusters that presents arbitrary shape, has non-uniform density and belongs to low dimensional hyperspaces. Furthermore, good performance of clustering algorithms is always in high demand because of execution time concerns [10].

Examples of clustering approaches include partitioning methods, hierarchical methods, density-based methods, model-based methods, graph-based methods, grid-based methods, among others [10]. However, for each approach, most of the clustering methods require the user to set parameters or make use of some prior knowledge about the data set in order for the algorithms to produce useful performance [10]. Hence, these non-automatic methods can be quite unreliable when the parameter tuning is not performed well (for each particular application).

Nosovskiy et al. [10] proposed a new algorithm, ADACLUS, for which the clustering procedure is performed based on local-adaptive influence function. That new influence function allows the algorithm to automatically discover clusters of arbitrary shape and different densities (i.e. it does not require any parameter pre-setting). The innovative feature of ADACLUS, in relation to previous clustering algorithm, is that the parameters of the influence function are not global and nor predefined [10].

In this section, firstly, we give some definitions necessary to the understanding of the ADACLUS algorithm. Subsequently, we present and explain each step of the ADACLUS algorithm.

A. Definitions and Parameters of ADACLUS

For calculating the influence of the data point \bar{P} at any point \bar{X} in the space, ADACLUS uses an adaptive influence function defined bellow, see (8).

$$f_{a,b}(\bar{P}, \bar{X}) = \begin{cases} 1 & \text{if } 0 \leq \rho_1(\bar{P}, \bar{X}) \leq a, \\ \frac{b - \rho_1(\bar{P}, \bar{X})}{b - a} & \text{if } a < \rho_1(\bar{P}, \bar{X}) \leq b, \\ 0 & \text{if } \rho_1(\bar{P}, \bar{X}) > b, \end{cases} \quad (8)$$

where $\rho_1(\bar{X}_1, \bar{X}_2) = \sum_{i=1}^d |x_1^i - x_2^i| \quad \forall \bar{X}_1, \bar{X}_2 \in \mathbb{R}^d$.

Although the metric ρ_1 is similar to the Euclidian metric, it results in faster execution time [10]. Parameters a and b denote maximum and minimum distances of data points \bar{P} , which are considered as still “near” and already “far”, respectively. In case of $a = b$, the function $f_{a,b}(\bar{P}, \bar{X})$ becomes the Square-Wave function as in (9).

$$f_{a,a}(\bar{P}, \bar{X}) = \begin{cases} 1 & \text{if } 0 \leq \rho_1(\bar{P}, \bar{X}) \leq a, \\ 0 & \text{if } \rho_1(\bar{P}, \bar{X}) > a \end{cases} \quad (9)$$

Given the influence function (8), the field function at any point \bar{X} in the space is defined as (10).

$$F(\bar{X}) = \sum_{i=1}^N f_{a_i, b_i}(\bar{P}_i, \bar{X}) \quad (10)$$

where N is the number of data points and a_i and b_i are used to calculate the influence of the data point \bar{P}_i at any space point \bar{X} . a_i and b_i are calculated for each \bar{P}_i in Step 5 of the ADACLUS algorithm.

Based on the definition of the field function F , a subset C of the data points is a cluster if: (i) the value of the influence function for each data point $\bar{P} \in C$ is greater than a threshold parameter $T \geq 0$, i.e. $F(\bar{P}) \geq T$; (ii) for all \bar{P}_i, \bar{P}_j there is a curve $x = x(s)$, $0 \leq s \leq 1$ for which $x(0) = p_i$, $x(1) = p_j$, $F(x(s)) \geq T \quad \forall s \in [0, 1]$. Notice that it is impossible to add any more data points to C without violating (i) and (ii).

In order to control the influence of local and global distribution of data points over calculations of a_i and b_i a magnifier parameter $G \in [0, 1]$ is defined. For $G = 1$, the clustering is purely local biased. For $G = 0$, the clustering algorithm is mainly global-biased, although it remains locally adaptive. If $0 < G < 1$, the clustering is performed based on a mixture of global and local information about the data set. The parameter G is calculated during Step 3 of the

ADACLUS algorithm if it is not overridden by the user.

B. The ADACLUS Algorithm

Let $\{\bar{P}_1, \bar{P}_2, \dots, \bar{P}_N\}$ be d -dimensional vectors for which $\bar{P}_i = \{p_i^1, p_i^2, \dots, p_i^d\}$ and let T and S set automatically to $T=1$ and $S=1000$. The steps of the ADACLUS algorithm are as follows [10], see Algorithm-1.

Algorithm-1:

Step 1. Quantization and integer-coding for each d -coordinate axes.

Step 2. Minimal distances calculation for each \bar{P}_i to the other data points according to ρ_i .

Step 3. Parameter G automatic determination.

Step 4. Calculation of the localization radius R .

Step 5. Determination of a_i and b_i for each data point \bar{P}_i .

Step 6. Computation of the field function in all data points using (10).

Step 7. Determination of the point-based clusters.

Step 8. Determination of the boundary-based clusters and boundary detection.

In Step 1, for each d coordinate axis, m_k and M_k are determined as $m_k = \min\{p_i^k, 1 \leq i \leq N\}$ and $M_k = \max\{p_i^k, 1 \leq i \leq N\}$, $\forall k \in \{1, 2, \dots, d\}$. Then, the d intervals $[m_k, M_k]$ are divided equally into S intervals Δ_j^k , $j \in \{1, 2, \dots, S\}$, where $S \in \mathbb{N}$. Later, each p_i^k is coded as an integer \tilde{p}_i^k such that $p_i^k \in \Delta_{\tilde{p}_i^k}^k$. This integer value is used in all further steps.

In Step 2, the minimal distances (MD) from each \bar{P}_i to all other data points is calculated.

In Step 3, the value of G is calculated according to the number of data points (G_0) and their distribution on the "screen" (G_1) is given by $G = G_0 + G_1 - G_0 G_1$, where

$$G_0 = \begin{cases} 1 & \text{if } N \leq S_{\min}, \\ 1 - \frac{N - S_{\min}}{S_{\max} - S_{\min}} & \text{if } S_{\min} \leq N \leq S_{\max}, \\ 0 & \text{if } N \geq S_{\max}, \end{cases} \quad (11)$$

$$G_1 = \begin{cases} 1 & \text{if } t \leq 1/\sqrt{d}, \\ 1 - \frac{t - (1/\sqrt{d})}{1 - (1/\sqrt{d})} & \text{if } 1/\sqrt{d} \leq t \leq 1, \\ 0 & \text{if } t \geq 1 \end{cases} \quad (12)$$

where N in (11) is the number of data points and d in (12) is the number of dimensions of the space.

The suggested values of S_{\min} and S_{\max} are $S_{\min} = 200$ and $S_{\max} = 1000$ [10]. The value of t is calculated as

$$t = std_{MD}/m_{MD}, \quad \text{where} \quad m_{MD} = \frac{1}{N} \sum_{i=1}^N MD(i) \quad \text{and}$$

$$std_{MD} = \sqrt{\sum_{i=1}^N (MD(i) - m_{MD})^2 / (N-1)}.$$

After the determination of G , in Step 4 the radius R of the neighborhood for which the local considerations will be performed is given as follows [10].

$$R = \min\{GR_{loc} + (1-G)R_{glob}, \sqrt{100m_{MD}/2}\} \quad (13)$$

$$R_{loc} = \min\{x : x \geq R_{glob}, g_{emp}^{MD}(x) = 0\} \quad (14)$$

$$R_{glob} = \min\{x : P_{emp}(MD < x) = 0.9\} \quad (15)$$

Let $g_{emp}^{MD}(x)$ be the corresponding frequency histogram of the subset of the whole MD distribution corresponding to MD values associated with the data points in the neighborhood of \bar{P}_i (i.e. $\rho_i(\bar{P}, \bar{P}_i) \leq R$ including \bar{P}_i). Let m_{MD}^i be its mean value, std_{MD}^i , its standard deviation and P_{emp} be a probability density function. Then, in Step 5, for each data point \bar{P}_i the values of a_i and b_i are calculated according to the neighborhood of \bar{P}_i with radius R as

$$a_i = (d_{i,1} + d_{i,2})/2, \quad \text{where}$$

$$d_{i,1} = \max\{x : x \leq m_{MD}^i, g_{emp}^{MD}(x) = 0\}$$

$$d_{i,2} = \max\{x : P^{MD}(MD < x) = 0.1\} \quad \text{and}$$

$P^{MD}(MD < x) = 0.1$ is 10%-fraction of the distribution $g_{emp}^{MD}(x)$. At last, $b_i = a_i + (4 - 2G)std_{MD}^i$.

Once all the ADACLUS parameters are calculated and after the field function is computed for each data point, the standard hill-climbing algorithm is used in order to obtain all the local maxima of F . Each local maxima will be selected as the center of a point based cluster of radius R .

For Step 8, the cluster boundaries are drawn for each cluster created in Step 7.

The ADACLUS algorithm has linear time complexity and presents total complexity is $O(dN)$ with respect to the number of data points N and the number of dimensions d [10].

V. ADAPTATIVE CLUSTERING PARTICLE SWARM OPTIMIZATION

It is intuitive that clustering processes can be directly applied to PSO in order to define different topologies for a swarm of particles. In our approach, we propose to periodically divide the swarm in clusters and allow these independent subswarms to operate similarly as Clan-PSO.

The clustering procedure is also a major contribution as it is performed based on velocity information ACVPSO (i.e. Adaptative Clustering based on Velocity Particle Swarm Optimization) rather than on position. To be fair, we also investigated another variation based on positional information of each particle of the swarm – ACPPSO (i.e.

Adaptative Clustering based on Position Particle Swarm Optimization).

We choose to use ADACLUS algorithm because, as described in Section 4, it does not need fine-tuning of parameters for each problem. By using ADACLUS to create clusters based on the particles' velocity, the algorithm may aggregate particles that are moving towards relatively closer directions of the search space.

In order to balance the exploration-exploitation abilities of clusters, we used in our approaches the mechanism of conference among clusters as proposed in Clan-PSO. According to this mechanism, the *gbest* of each cluster is selected and another PSO algorithm is executed with the resultant swarm of leaders using the star topology. Differently from Clan-PSO, in our approach, after a fixed number of iterations of the leaders' conference (I_{glob}), all the particles "return" to the main swarm and ADACLUS is performed to create new clusters according to the new velocities (or positions) of particles. The created clusters perform a local search during a fixed number of iterations (I_{loc}). After the local search process, in each subswarm, is completed the conference mechanism described previously starts all over. Those steps are preformed in this sequence until the stop criterion is reached (e.g. maximum number of iterations or target performance).

Once the neighborhood is created dynamically for each particle, it is possible that some clusters comprise only few particles (e.g. two particles). That can occur especially when the velocity of all particles is close to zero, for which the particles' velocity are densely grouped around the origin. Hence, to avoid premature convergence of those clusters with few particles, we use GCPSO algorithm in the local search process to increase the diversity. This approach is based on the suggestion presented in NichePSO [11]. Also from NichePSO, we adapt the technique to identify niches. That is, to monitor changes in the fitness of individual particles and changes in the *gbest* of each cluster. In our approach, if the standard deviation of that value after three iterations of GCPSO is lower than a threshold, the search process in this specific cluster is stopped. After all the local search processes are finished, another conference (of leaders) is performed.

The conference search-mode must guarantee the particles to converge to the best region of the search space found so far by all the clusters in order to exploit that region. For this, we adopted CPSO as the search algorithm for the conference process intending to control the growth of the velocity and to avoid rapid convergence to local optima. All these ideas put together are deemed to strike an appropriate (and necessary) balance between exploration and exploitation.

The proposed new PSO algorithm, that includes the hybrid suggested topology is given as follows, see Algorithm-2.

Algorithm-2:

Step 1. Create and initialize the swarm. For the

initial velocity, we assign random vectors $\vec{v}_i = (v_i^1, v_i^2, \dots, v_i^d)$, where v_i^k is uniformly distributed in $[0,1]$.

Step 2. **While** a stop criterion is not satisfied **Begin**.

Step 3. Crete clusters using ADACLUS based on the particles' velocity information.

Step 4. Run GCPSO for each cluster up to a fixed number of iterations I_{loc} or $std_{gbest} < \delta \quad \forall i \in \{1, 2, \dots, c\}$ where c is the number of created clusters.

Step 5. Select the *gbest* of each cluster and perform a search process using CPSO up to a fixed number of iterations I_{glob} .

Step 6. **End While**.

Step 7. **Return** the best solution found by the algorithm.

VI. EXPERIMENTS

To assess the effectiveness of our proposal, we have carried out two different analyses (experiments). In the first one, we compared the results obtained using the velocity information and the position information as means to creation of the clusters. In the latter, we compared the best results using the velocity information with those results achieved by other recent proposed PSO approaches.

In all the experiments performed and presented in this paper, we used the parameter values proposed in the original white paper of each technique utilized. That is, the parameter values used for CPSO were $c_1 = c_2 = 2.05$. For GCPSO, we used $c_1 = c_2 = 1.2$, $\omega = 0.7$, $s_c = 15$, $f_c = 5$. We assumed $S = 1000$, $T = 1$ for ADACLUS. δ is equal to 10^{-4} . All the simulations we performed using 30 particles and 10,000 iterations and each experiment were repeated 30 times. In Table I, we present the general parameters for all benchmark functions used in this paper.

In the first experiment, we investigated the influence of the number of consecutive iterations performing local and global searches (i.e. I_{loc} and I_{glob}) in our proposals. The results are shown in Tables II, III, IV, V and VI.

Table II presents the results for ACVPSO and ACPPSO for *Ackley* function varying I_{loc} and I_{glob} . For the ACVPSO, we achieved the best performance with relatively low number for I_{loc} and I_{glob} , i.e. $I_{loc} = I_{glob} = 100$. We also achieved good performance for ACPPSO. However, in this case the I_{loc} must be higher than I_{glob} .

TABLE II
COMPARATIVE RESULTS BETWEEN ACVPSO AND ACPPSO FOR *ACKLEY*

Local Global	ACVPSO		ACPPSO	
	Mean	Std.	Mean	Std.
50 50	9.87848E-5	2.0360E-5	1.01218E-4	2.3904E-5
100 100	3.47028E-6	9.3732E-7	8.25514E-6	2.0071E-5
500 500	18.3048275	4.3375215	17.5428218	5.7463924
100 500	17.6191456	5.6173360	17.2880891	6.0545573
500 100	1.31253350	4.9162529	4.82700E-6	4.8963E-6

700 300	11.0516774	9.6760775	14.7383059	8.1154987
300 700	19.5817984	0.7589812	18.5722473	4.2741702
1000 1000	19.7574628	0.1116475	19.0428387	3.5497917

Table III presents the results for ACVPSO and ACPPSO for *Griewank* function varying I_{loc} and I_{glob} . One may notice that the best results were obtained again for relatively small values of I_{loc} and I_{glob} in both cases. The best performance for was achieved by using ACVPSO.

TABLE III
COMPARATIVE RESULTS BETWEEN ACVPSO AND ACPPSO FOR *GRIEWANK*

Local Global	ACVPSO		ACPPSO	
	Mean	Std.	Mean	Std.
50 50	0.01368152	0.0213230	0.02288037	0.0243154
100 100	0.04225285	0.0547522	0.03326679	0.0379652
500 500	0.06159859	0.0519108	0.07724284	0.0636622
100 500	0.07513396	0.0791302	0.04670543	0.0427355
500 100	0.03743177	0.0364332	0.03150286	0.0383188
700 300	0.07616084	0.0848358	0.06011091	0.0746276
300 700	0.07164149	0.0911547	0.08809375	0.1055870
1000 1000	0.08773713	0.1224160	0.08167104	0.1191402

Table IV presents the results for ACVPSO and ACPPSO for *Rastrigin* function varying I_{loc} and I_{glob} . One may notice that the ACVPSO and ACPPSO achieved the best performance with a relatively low number for I_{loc} and I_{glob} , i.e. $I_{loc} = I_{glob} = 100$. Furthermore, the ACVPSO outperformed the ACPPSO.

TABLE IV
COMPARATIVE RESULTS BETWEEN ACVPSO AND ACPPSO FOR *RASTRIGIN*

Local Global	ACVPSO		ACPPSO	
	Mean	Std.	Mean	Std.
50 50	13.9697145	6.4058471	25.8108713	16.087941
100 100	5.63851575	4.7534524	18.6156118	18.310537
500 500	35.2093951	16.797269	47.5689091	20.952522
100 500	36.3329035	17.878464	46.4152131	23.702880
500 100	12.5909786	17.662477	17.9518238	13.370210
700 300	23.6551005	12.151288	31.6155615	17.049988
300 700	50.8657426	18.474188	63.6992287	19.603110
1000 1000	71.1420983	21.398619	78.2563935	24.455646

Table V presents the results for ACVPSO and ACPPSO for *Rosenbrock* function varying I_{loc} and I_{glob} . For the ACVPSO, the best performance was achieved with a relatively low number for I_{loc} and I_{glob} , i.e. $I_{loc} = I_{glob} = 100$. We also achieved a good performance for ACPPSO. However, in this case the I_{loc} must be higher than I_{glob} .

TABLE V
COMPARATIVE RESULTS BETWEEN ACVPSO AND ACPPSO FOR *ROSENBRICK*

Local Global	ACVPSO		ACPPSO	
	Mean	Std.	Mean	Std.
50 50	12.8136555	13.511760	22.1490633	27.754603
100 100	7.82253146	4.9246199	13.5330104	19.976061
500 500	26.6983826	31.942378	23.7480774	27.931596
100 500	29.1888959	32.495576	21.4483522	25.562269
500 100	10.4447307	14.496972	12.7653737	20.056452

700 300	21.4693499	27.155320	14.8284002	17.210543
300 700	26.2557258	32.161828	22.3924121	25.852017
1000 1000	30.0444956	32.634385	34.4761854	33.584193

Table VI presents the results for ACVPSO and ACPPSO for *Schwefel* function varying I_{loc} and I_{glob} . In this case, the algorithm behaved differently. The best results were achieved when $I_{glob} > I_{loc}$. However, as for all other benchmark functions, the best performances were achieved by ACVPSO.

TABLE VI
COMPARATIVE RESULTS BETWEEN ACVPSO AND ACPPSO FOR *SCHWEFEL 1.2*

Local Global	ACVPSO		ACPPSO	
	Mean	Std.	Mean	Std.
50 50	0.02331672	0.0062106	0.02257332	0.0070632
100 100	8.85841E-4	0.0011753	5.55114E-4	5.4899E-4
500 500	9.5989E-13	1.401E-12	3.2876E-12	6.011E-12
100 500	4.1228E-13	9.538E-13	3.0853E-12	3.423E-12
500 100	8.78135E-4	9.1922E-4	6.65617E-4	4.9869E-4
700 300	2.3462E-12	3.688E-12	8.1988E-12	1.553E-11
300 700	8.8209E-11	1.440E-10	5.1163E-10	8.456E-10
1000 1000	6.61705E-9	1.0355E-8	7.24183E-8	2.2306E-7

In the second experiment, we compared the best results for ACVPSO and ACPPSO for all the same benchmark functions with the results obtained from other PSO approaches. The results are shown in Tables VII, VIII, IX, X and XI for *Ackley*, *Griewank*, *Rastrigin*, *Rosenbrock* and *Schwefel 1.2*, respectively.

TABLE VII
COMPARATIVE RESULTS BETWEEN PSO APPROACHES FOR *ACKLEY*

Technique	Mean	Std.
CPSO (Local)	17.58910	1.0264
CPSO (Global)	17.66280	1.0232
Clan-PSO	16.20549	7.8440
ACPPSO	4.82700E-6	4.8963E-6
ACVPSO	3.47030E-6	9.3732E-7

TABLE VIII
COMPARATIVE RESULTS BETWEEN PSO APPROACHES FOR *GRIEWANK*

Technique	Mean	Std.
CPSO (Local)	0.0009	0.0005
CPSO (Global)	0.0308	0.0063
Clan-PSO	1.00061	0.0004
ACPPSO	0.02288037	0.0243154
ACVPSO	0.01368150	0.0213230

TABLE IX
COMPARATIVE RESULTS BETWEEN PSO APPROACHES FOR *RASTRIGIN*

Technique	Mean	Std.
CPSO (Local)	144.815500	4.406600
CPSO (Global)	140.487600	4.853800
Clan-PSO	5.011460	4.015000
ACPPSO	18.6156118	18.3105370
ACVPSO	5.6385157	4.7534524

TABLE X
COMPARATIVE RESULTS BETWEEN PSO APPROACHES FOR *ROSENBRICK*

Technique	Mean	Std.
CPSO (Local)	12.6648000	1.230400
CPSO (Global)	8.1579000	2.783500
Clan-PSO	3.2855100	3.493000

ACPPSO	12.7653737	20.056452
ACVPSO	7.6524475	5.8755074

TABLE XI

COMPARATIVE RESULTS BETWEEN PSO APPROACHES FOR SCHWEFEL 1.2

Technique	Mean	Std.
CPSO (Local)	0.1259	0.0178
CPSO (Global)	0.0	0.0
Clan-PSO	2.0100E-13	3.900E-13
ACPPSO	3.0853E-12	3.423E-12
ACVPSO	4.1230E-13	9.538E-13

One can note that ACVPSO outperformed all the other PSO approaches for the *Ackley* function. The ACVPSO and Clan PSO achieved similar performances and outperformed the other approaches in *Rastrigin* and *Schwefel 1.2* functions. Although, our approach has not achieved the best performance for the *Rosenbrock* function, the obtained results for ACVPSO were not bad at all. The same has happened for the *Griewank* function, as the obtained results beat the ones produced by Clan-PSO approach.

VII. CONCLUSION

In this paper, we investigated the use of clustering algorithms in order to create the particles' neighborhoods in a cluster topology based on the particles' velocity information (rather than spatial position). In conjunction with inspiration of functioning drawn from Clan-PSO, we most certainly afforded to the introduced approach much required flexibility and some performance.

In general, better results were achieved for multimodal functions (e.g. *Rastrigin* and *Griewank*) with $I_{loc} = I_{glob}$; and ACVPSO outperformed the ACPPSO in all the cases. Overall comparisons, indicate that the proposed approach is flexible regarding application domain as produces highly competitive results.

Based on the first and second experiments (refer to Table II and Fig. 3, respectively), we can conclude that the performance of the cluster topology using the velocity information is generally better than what is achieved by algorithms that use particles' position information.

As future works, we will try to equip this new approach with the ability to automatically regulate the quantity of local and global iterations, according to each problem at hand.

ACKNOWLEDGMENT

The authors thank the Polytechnic School of Pernambuco, University of Pernambuco, FACEPE (Pernambuco State Research Foundation) and CNPq (Brazilian National Research Council) for the financial support for this paper.

REFERENCES

[1] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in Proc. of the IEEE Int. Conf. on Neural Networks. Piscataway, NJ: IEEE Service Center, 1995, pp. 1942–1948.

[2] P.N. Suganthan, "Particle Swarm Optimiser with Neighborhood Operator," In Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1958–1962, 1999.

[3] D.F. Carvalho and C.J.A. Bastos-Filho, "Clan Particle Swarm Optimization," IEEE Congress on Evolutionary Computation, 2008, pp. 3044 - 3051, 1999.

[4] M. Dorigo, G. Di Caro, "The ant colony optimization metaheuristic," McGraw-Hill, New York (1999), pp. 11–32.

[5] Y. Shi and R. Eberhart, "A Modified Particle Swarm Optimizer," Proceedings of the 1998 IEEE Congress on Evolutionary Computation, Anchorage, AK, pp. 69-73.

[6] I. C. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," Information Processing Letters, Volume 85, Issue 6, pp. 317-325, March 2003.

[7] M. Clerc and J. Kennedy (2002). The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation, 6(1): pp. 58-73.

[8] R. Eberhart and Y. Shi, "Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization," Proceedings of the 2000 IEEE Congress on Evolutionary Computation, Volume 1, pp. 84-88.

[9] F. van den Bergh, A. P. Engelbrecht, "A new locally convergent particle swarm optimizer," Proceedings of the IEEE Conference on Systems, Man and Cybernetics, pp. 96-101, October 2002.

[10] G. V. Nosovskiy, L. Dongquan, O. Sourina, "Automatic clustering and boundary detection algorithm based on adaptive influence function," Pattern Recognition, Volume 41, Issue 9, pp. 2757-2776, September 2008.

[11] R. Brits, A.P. Engelbrecht, F. van den Bergh, "Locating multiple optima using particle swarm optimization," Applied Mathematics and Computation, Volume 189, Issue 2, pp. 1859-1883, June 2007.

[12] C. Bastos-Filho, F.B.de Lima Neto, A.J.C. C. Lins, A.I. S. Nascimento, M.P. Lima, "A novel search algorithm based on Fish School Behavior," Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, pp. 2646-2651, October 2008.