Particle Swarms for Multimodal Optimization

Ender Özcan, Murat Yılmaz

Yeditepe University, Department of Computer Engineering,
34755 Kadıköy/İstanbul, Turkey
eozcan/myilmaz@cse.yeditepe.edu.tr

Abstract. In this paper, five previous Particle Swarm Optimization (PSO) algorithms for multimodal function optimization are reviewed. A new and a successful PSO based algorithm, named as CPSO is proposed. CPSO enhances the exploration and exploitation capabilities of PSO by performing search using a random walk and a hill climbing components. Furthermore, one of the previous PSO approaches is improved incredibly by means of a minor adjustment. All algorithms are compared over a set of well-known benchmark functions.

1 Introduction

Inspiring from the swarms in nature, such as; birds, fish, etc., Kennedy and Eberhart [7] proposed a population based algorithm called Particle Swarm Optimization (PSO). PSO combines *cognition only model* that values solely the self-experience and *social only model* that values solely the experience of neighbors. A particle encodes a candidate solution to a problem at hand. The algorithm uses a set of particles flying over a search space and moving towards a promising area to locate a global optimum. However, there are a set of problems requiring discovery of equal quality candidate solutions, so that, a user could make a choice in between them. In some problems, local optima, or a set of solutions with a predetermined quality levels can also be requested. Multimodal optimization problems represent such class of problems in which the researchers are interested. Different PSO algorithms have been already proposed for solving multimodal problems. These algorithms are mostly based on existing approaches used in the evolutionary algorithms for multimodal optimization.

Most of the real world problems carry multimodal characteristics; hence developing efficient algorithms for multimodal optimization problems is still a research area. Previous approaches can be categorized as *iterative* and *subpopulation* methods [8]. In the iterative methods, the algorithm is applied several times consecutively to locate each optimum. In the subpopulation methods, the population is divided into parts to search optima simultaneously. In this paper, the previous PSO algorithms for multimodal optimization based on subpopulation model are compared to a proposed Particle Swarm Optimizer with craziness and hill climbing, named as CPSO. Additionally, a previous niching PSO approach is modified, yielding an improved performance. All algorithms are described in Section 2 and 3. The experimental results are presented in Section 4. Finally, the conclusions are provided in Section 5.

2 PSO Systems for Multimodal Function Optimization

In a PSO system based on *inertia weight* [6], particles representing candidate solutions start their flight from random locations in a search landscape. At each step, a particle updates its velocity to move to another location based on Eq.(1) and (2). The flight is influenced by a *fitness function* that evaluates the quality of each solution.

$$v_{id}(t+1)=w\, v_{id}(t)+c_1\, r_1(t)\, (y_{id}(t)-x_{id}(t))+c_2\, r_2(t)\, (y_{gd}(t)-x_{id}(t)) , \quad (1)$$

$$x_{id}(t+1)= x_{id}(t)+ v_{id}(t+1) . \quad (2)$$

where $x_{id}(t)$ is the position of the i^{th} particle at time t on dimension d, v is the velocity, w is the inertia weight, c_1 and c_2 are constant values, r_1 and r_2 are uniform random numbers in [0,1], y_i is the i^{th} particle's best position (generating the best fitness) that has been found so far, and y_g is the best position visited by the neighbors. Generally, the neighborhood is chosen as the whole population for global optimization. If c_1 is set to 0 (and $c_2 \neq 0$), the PSO system turns into the social only model and if c_2 is set to 0 (and $c_1 \neq 0$), then the system becomes the cognition only model. In this paper, five multimodal PSO algorithms are discussed that are extended from the generic approach: *Species-based PSO* (SPSO), *Niching PSO* (NichePSO), *nbest PSO* (nbestPSO), *Unified PSO* (UPSO) and *Parallel Vector-Based PSO* (PVPSO).

Li's [8] *Species-based PSO* gathers the *similar* particles into the sub-swarms called *species*. As a similarity measure Euclidean distance is used. SPSO requires an additional parameter called *species radius*; r_s. The best fit particle in a species is called the *species seed*, and the *boundary* of a species is the circle having the radius of r_s around this seed. The particles in the entire swarm move within their own species at each iteration. Then, they are evaluated and the species are redefined. The multiple optima are maintained in a parallel manner. The convergence rate of the algorithm is enhanced by the communication of particles in the swarm through the PSO algorithm and the reconstruction of the species.

Brits, Engelbrecht and van den Bergh [4] proposed the *nbestPSO*. This method redefines the neighborhood best position to increase the diversity during the information sharing between particles. For each particle i, k nearby particles are determined, and the neighborhood best position y_{gi} is calculated as the center of mass of the best positions visited by these k particles. In Eq.(1), y_{gi} replaces y_g. Then the same velocity update equation in Eq.(2) is used. Increasing, decreasing and constant k values are analyzed by the researchers. The results show that linearly decreasing k value yields the best performance.

The *Unified PSO*, introduced by Parsopoulos and Vrahatis [13], aims to bring a balance to the global and local variants of PSO. The algorithm requires local and global neighborhoods to be defined. In this algorithm, the velocity update equation ($U_i^{(k+1)}$) is changed and divided into local ($L_i^{(k+1)}$) and global ($G_i^{(k+1)}$) parts. A particle samples two different velocities using two different velocity update PSO equations based on the constriction PSO model, where the constriction factor X controls the

velocity's magnitude [6]. The newly introduced *unification factor*; $u \in [0,1]$ determines the effect of the global and local information:

$U_i^{(k+1)} = u\, G_i^{(k+1)} + (1-u)\, L_i^{(k+1)}$, where $U_i^{(k+1)}$ is the new location of the i^{th} particle after the k^{th} iteration. Additionally, a normally distributed random parameter is also introduced to be multiplied with either $G_i^{(k+1)}$ or $L_i^{(k+1)}$, yielding two different models that supports mutation.

The *Parallel Vector-Based PSO* (PVPSO), introduced by Schoeman and Engelbrecht [14], uses a set of vector operations to form niches in the search space. PVPSO performs better than their previous approach the Vector-Based PSO (VBPSO). In PVPSO, the initial niches are identified as in VBPSO, but all particles are evaluated simultaneously. The velocity update is done using the personal best and the best neighborhood positions. A sub-swarm may absorb the other particles that come close by and/or merge with another one.

2.1 The Niching Particle Swarm Optimizer (NichePSO)

Brits, Engelbrecht and van den Bergh proposed an algorithm as presented in Fig.1 to locate the multiple optima using a particle swarm based algorithm, referred as *NichePSO* [5]. The initial swarm, called as the main swarm is generated by uniformly distributing particles over the search space. The quality of the particles is monitored during the iterations. If a particle's fitness does not change for some epochs, its position is set to be a candidate solution. Then, this particle is removed from the main swarm and a new sub-swarm is generated. As the algorithm proceeds, the main swarm loses its members as the new sub-swarms are created. Dynamically generated sub-swarms are expected to locate all global and local optima in parallel.

NichePSO

Initialize particles in the main swarm
Repeat
1. Train particles in the main swarm using a single iteration of the cognition only model
2. Update fitness of each particle in the main swarm
3. For each subswarm:
    a. Train subswarm particles using a single iteration of the GCPSO algorithm
    b. Update fitness of each particle
    c. Update subswarm radius
4. If possible, merge subswarms
5. Allow subswarms to absorb any particles from the main swarm that moved into it
6. Search main swarm for any particle that meets the partitioning criteria – If found, create a new subswarm with this particle and its closest neighbor

Until stopping criteria are met

Fig. 1. Pseudocode for the NichePSO algorithm

The algorithm of Løvbjerg et al. [10] is adapted for improving swarm diversity. If the swarm size is small, then PSO algorithm has a disadvantage of getting stuck at a

position when $x_i \cong y_i \cong y_g$, where velocity might approach to zero. Therefore, NichePSO uses Van den Bergh's GCPSO algorithm [1] to prevent sub-swarms from halting. The initial swarm is vital for the success of the NichePSO, hence Faure-sequences are used to distribute the initial particles uniformly over the search space. If a particle does not belong to a niche, and the variance of its fitness is below a threshold for some epochs, then a niche is created around it. These niches may merge, if the distance between the best particles in them is less than some value μ or absorb the other particles which do not belong to a niche. NichePSO is implemented as described in [5]. It is observed that the niche radius may increase, spanning the whole search space and causing most of the particles to converge to a single optimum. In this paper, a modified version, referred as *mNichePSO* is proposed to prevent this type of behavior. Simply, the niche radius size is not allowed to exceed a maximum value.

3 PSO with Craziness and Hill Climbing (CPSO)

In most of the algorithms used for optimization, the balance between exploration and exploitation is vital for success. The proposed CPSO algorithm (Fig. 2) for multimodal function optimization uses a random walk component and a hill climber to enhance the exploration and exploitation capabilities of PSO, respectively.

CPSO

Initialize particles

Repeat

1. On the first and every m epochs, construct sub-swarms according to their geographical positions, with a neighborhood size n.
2. For each particle compute 2 candidate positions:
   a. Use the original PSO, where y_g is the sub-swarm's best.
   b. if (the fitness variance of a particle or particles in a subswarm for k epochs is smaller than a *variance* threshold)
      then
         use the original PSO, where y_g is the sub-swarm's best
      else
         generate a random position using Eq.(5)
3. Compute the fitness values of these candidate positions. Choose the position with the better fitness as particles' current positions. If the random position produces a better fitness, set the particles' velocities using Eq.(6).
4. If (the fitness variance of a particle for p epochs is smaller than a *variance* threshold)
   then
      reset the velocity of the particle using Eq.(6).

Until stopping criteria met

Fig. 2. Pseudocode for the CPSO algorithm

In this algorithm, the main swarm is divided into sub-swarms of size n according to their geographical positions. Starting from the first particle, for each particle which

does not belong to a sub-swarm, the nearest (n−1) particles, which also do not belong to a sub-swarm, are detected. At every m epochs, the sub-swarms are rearranged according to their current geographical positions. This action provides a type of communication and information diffusion between particles, since a local best value might change within a neighborhood. Each particle generates two candidate positions, denoted by x^1 and x^2, at each epoch. Let v^1 and v^2 denote velocities for the related candidate positions. In Eq.(3), y_{gi} is the best position visited so far within the neighborhood of the i^{th} particle. The parameters $maxx_d$ and $minx_d$, in Eq.(5), are the limits of the search space at dimension d. The first candidate position is computed using Eq.(3) and (4), and the second one is computed using Eq.(5) during the initial moves. For the candidate positions x^1 and x^2, each particle makes a decision based on the fitness values. A particle moves to the position that generates a better fitness.

$$v^1_{id}(t+1)=w\, v^1_{id}(t)+c_1\, r_1(t)\, (y_{id}(t)-x_{id}(t))+c_2\, r_2(t)\, (y_{gid}(t)-x_{id}(t))\,, \tag{3}$$

$$x^1_{id}(t+1)=x^1_{id}(t)+v^1_{id}(t+1)\,, \tag{4}$$

$$x^2_{id}(t+1)= minx_d + r_3\, (maxx_d-minx_d)\,, \tag{5}$$

Moving to a better candidate position can be considered as a hill climbing step. A single meme consisting of two phases is used: *sampling* and *acceptance*. As a sampling technique either a random walk (craziness) or the PSO algorithm itself is invoked, depending on the mode of operation as described in Fig. 2. As an acceptance strategy, only improving moves are admitted. If the position x^2 is chosen, then the previous velocity becomes useless. Hence, a new velocity has to be assigned to the particle. Eq.(6) is used for that purpose.

$$v_{id}(t+1)= \alpha\, maxv\, r_5(t+1)\,, \tag{6}$$

where α is a constant number, $maxv$ is the limit for the velocity of the particles, and $r_5(t)$ is a uniform random number in [-1,1] at time t. If the *variance* of the fitness for the last p epochs is smaller than a threshold value, all particles in a sub-swarm stop making random moves. The mode of operation switches to a refined search. Particles generate two velocities invoking the Eq.(3) twice, yielding two candidate positions and the search continues as described in Fig. 2. CPSO algorithm introduces the following parameters: n, k, p, m, *variance* (threshold) and α.

4 Experiments

The runs are performed on a 2 GHz, Windows 2003 operating system with 512 MB of memory. A Matlab application is implemented for the experiments, available at http://cse.yeditepe.edu.tr/ARTI/projects/cpso. Each experiment is repeated 50 times. A run is terminated either the maximum number of evaluations is exceeded or all required global optima are found within a fitness range of 0.00005.

4.1 Experimental Setup and Comparison Criteria

Well-known benchmark functions are used during the experiments as presented in Table 1. The initial experiments are performed for obtaining the best set of parameters for CPSO. Then, seven different multimodal PSO algorithms (SPSO, NichePSO, nbestPSO, CPSO, UPSO, mNichePSO, PVPSO) are tested on 10 different benchmark functions (F1-F10). The swarm size is chosen as 30 and 50, for functions F1-F4 and F5-F10, respectively. The problem dimension is one for the functions F1-F4, two for the functions F5-F10. For a fair comparison, the maximum number of evaluations is limited to 15,000 for F1-F4, and 25,000 for F5-F10. In UPSO, X is used as 0.729, and c_1 and c_2 are set to 2.05. For the rest of the algorithms, the inertia weight is linearly decreased from 0.8 to 0.6, and c_1 and c_2 are set to 1.5 for a stable PSO. More discussions on the choice of parameters can be found in [6], [8], [11] and [12].

Table 1. Benchmark functions used during the experiments; *minx* and *maxx* indicate the lower and upper bound for each dimension, #gl. and #lo. indicate the total number of global and local optima, respectively, for the corresponding function

lb.	Function	*minx, maxx*	#gl.	#lo.	Source(s)
F1	$y=1-\sin(5\pi x)^6$	0.0, 1.0	5	0	[8,5,1]
F2	$y=1-\exp(-2\log(2)((x-0.1)/0.8)^2)\sin(5\pi x)^6$	0.0, 1.0	1	4	[8,5]
F3	$y=1-\sin(5\pi(x^{(3/4)}-0.05))^6$	0.0, 1.0	5	0	[8,5,1]
F4	$y=1-(\exp(-2\log(2)*((x-0.08)/0.854)^2)\sin(5\pi\,(x^{(3/4)}-0.05))^6)$	0.0, 1.0	1	4	[8,5,1]
F5	$z=(x^2+y-11)^2-(x+y^2-7)^2$	–10.0, 10.0	4	0	[8,5]
F6	$z=\sin(2.2\,\pi\,x+\pi/2)((\,2-\lvert y\rvert\,)/2)((\,3-\lvert x\rvert\,)/2)+\sin(0.5\pi y^2+\pi/2)((\,2-\lvert y\rvert\,)/2)((\,2-\lvert x\rvert\,)/2)$	–2.0, 2.0	4	8	[16,17]
F7	$z=\cos(x)^2+\sin(y)^2$	–4.0, 4.0	6	0	[12]
F8	$z=(\cos(2x+1)+2\cos(3x+2)+3\cos(4x+3)+4\cos(5x+4)+5\cos(6x+5))(\cos(2y+1)+2\cos(3y+2)+3\cos(4y+3)+4\cos(5y+4)+5\cos(6y+5))$	–2.0, 2.5	2	38	[8,15]
F9	$z=(y^2-4.5y^2)xy-4.7\cos(3x-y^2(2+x))\sin(2.5\pi x)+(0.3x)^2$	–1.2, 1.2	1	9	[13]
F10	$z=4x^2-2.1x^4+(1/3)x^6+xy-4y^2+4y^4$	–1.9, 1.9	2	4	[13]

In SPSO, the species radius is chosen as (maxx–minx)/20. In NichePSO, the parameters μ and the variance are set to 0.001 and 0.0001, respectively [5]. Additionally, in mNichePSO, the maximum niche radius is set to (maxx–minx)/10. In nbestPSO, the neighborhood size is linearly decreased from 6 to 2. To evaluate the effect of the maximum velocity, each algorithm is investigated with two different values: (maxx–minx)/20 and (maxx–minx)/2, where each algorithm is labeled as *algorithm_abbriviation*-20 and *algorithm_abbriviation*-2, respectively.

In the multimodal optimization problems, not only multiple global optima having equal quality are to be searched, but also a predetermined set of local optima might be required. Therefore, it is difficult to evaluate and compare different algorithms.

Global peak ratio (*gpr*) is defined as the ratio of the average number of global optima found to the total number of global optima. *Local peak ratio* (*lpr*) is defined similarly using the local optima. *Global success consistency ratio* (*gscr*) denotes the proportion of the runs in which all global optima are discovered to the total number of runs, and *local success consistency ratio* (*lscr*) denotes the proportion of the runs in which all local are discovered to the total number of runs. The *overall success rate* (*osr*) is used as a single comparison criterion to evaluate all these aspects at once as shown in Eq. (7). If there are no local optima, *lpr* and *lscr* are set to 0, and the divisor to 2. Consequently, *osr* values are in the range [0,1] and a higher ratio indicates a better performance.

$$osr = \frac{(gpr + gscr + lpr + lscr)}{4} \tag{7}$$

4.2 Parameter Tuning for CPSO

324 different parameter sets are tested based on the combination of the following values for each CPSO parameter with *maxv*=(*maxx*–*minx*)/20:

- The sub-swarm size; $n \in \{2, 3, 4\}$
- The number of epochs $k \in \{3, 5, 7\}$ (step 2 (b) in Fig. 2)
- The number of epochs $p \in \{3, 5, 7\}$ (step 4 in Fig. 2)
- The *variance* (threshold)$\in \{0.0001, 0.01\}$ (step 2 (b) and 4 in Fig. 2)
- The number of epochs $m \in \{5,7,10\}$ for the sub-swarm reconstruction
- The velocity reset constant $\alpha \in \{0.001, 0.01\}$ (step 3 and 4 in Fig. 2, Eq. (6))

The performance of CPSO with each parameter set is compared with respect to the average *osr* over all benchmark functions. The results are summarized in Table 2. The best parameter set contains: n=2, k=3, p=3, *variance*=0.01, m=5 and α=0.01. It seems that relatively low values for n, k, p, and m and relatively high values for *variance* and α are *good* initial choices in a CPSO.

Table 2. The parameter sets that rank top ten based on their performances

rank	***variance***	***n***	***k***	***p***	***m***	***α***
1	0.01	2	3	3	5	0.01
2	0.01	2	3	5	7	0.01
3	0.01	2	3	3	7	0.01
4	0.01	2	3	7	5	0.01
5	0.01	2	3	5	5	0.01
6	0.01	2	3	5	5	0.001
7	0.01	2	3	7	10	0.01
8	0.01	2	3	3	10	0.01
9	0.01	2	3	7	7	0.01
10	0.01	2	3	3	5	0.001

4.3 Experimental Results

During the experiments, none of the algorithms is capable of finding all optima on all functions as summarized in Table 3. The maximum velocity choice can affect the performance of an algorithm considerably. NichePSO, mNichePSO, CPSO and PVPSO are more sensitive to the maximum velocity as compared to the others. On average, NichePSO, mNichePSO, CPSO algorithms perform better with a relatively low maximum velocity. PVPSO is the only algorithm that performs better with a relatively high maximum velocity. This choice does not generate a significant performance variance for SPSO, nbestPSO and UPSO.

Table 3. Average *osr* of each algorithm over all runs for each benchmark function. The best values (algorithms) for the benchmark functions are marked with the bold entries.

Algorithm	**F1**	**F2**	**F3**	**F4**	**F5**	**F6**	**F7**	**F8**	**F9**	**F10**	**avr.*osr*.**	**stdev.**
SPSO-2	0.98	0.82	0.96	0.84	**1.00**	0.64	0.97	0.58	0.65	0.59	0.80	0.17
SPSO-20	0.99	0.99	**1.00**	0.94	0.95	0.48	0.72	0.27	0.64	0.82	0.78	0.25
NichePSO-2	0.45	0.69	0.40	0.64	0.13	0.10	0.15	0.13	0.59	0.14	0.34	0.24
NichePSO-20	0.97	**1.00**	0.94	0.96	0.19	0.21	0.45	0.13	0.66	0.45	0.60	0.35
nbest-2	0.71	0.85	0.77	0.80	0.89	0.47	0.72	0.41	0.52	0.60	0.67	0.17
nbest-20	0.81	0.93	0.74	0.84	0.90	0.43	0.74	0.31	0.46	0.58	0.67	0.22
CPSO-2	0.96	0.78	0.83	0.78	0.28	0.56	0.98	0.17	0.56	0.53	0.64	0.27
CPSO-20	0.96	0.87	0.90	0.90	**1.00**	**0.66**	**1.00**	0.52	0.63	0.62	0.81	0.18
UPSO-2	0.72	0.70	0.66	0.70	0.81	0.47	0.64	0.51	0.56	0.50	0.63	0.11
UPSO-20	0.72	0.81	0.68	0.74	0.88	0.54	0.55	0.48	0.45	0.54	0.64	0.15
mNiche-2	0.70	0.69	0.61	0.67	**1.00**	0.54	0.91	**0.62**	**0.75**	0.64	0.71	0.14
mNiche-20	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**0.66**	0.99	0.14	0.47	**0.91**	**0.82**	0.30
PVPSO-2	0.99	0.93	0.98	0.97	0.45	0.57	0.79	0.39	0.55	0.62	0.72	0.23
PVPSO-20	**1.00**	**1.00**	**1.00**	**1.00**	0.21	0.38	0.46	0.20	0.35	0.71	0.63	0.35

The proposed modification to NichePSO, mNichePSO as described in Section 2.1, delivers a significantly better performance compared to the original one considering the average *osr*. Furthermore, mNiche-20, CPSO-20 and SPSO-2 are the top three algorithms in that order with respect to the average *osr*. PVPSO-20 is as successful as mNichePSO-20 for locating the multiple optima in the 2D search landscapes, while its performance deteriorates extremely for the 3D search landscapes.

The particle positions at the end of a sample run of a PSO algorithm for F6 are illustrated in Fig. 3. SPSO is good at locating the required optima. Mostly, at end of the runs, the species were evenly distributed around these optima. NichePSO provides good separation of niches, but, sometimes, all niches might merge into a single one. Preventing the niche radius to grow, as in mNichePSO, yields a better diversity. The nbestPSO produces a poor convergence rate. The proposed CPSO algorithm is very successful in locating the global optima as compared to the local optima (Fig. 3-4). Although, UPSO performs poorly, the parameters provide an effective way to derive

the algorithm to locate any optima.

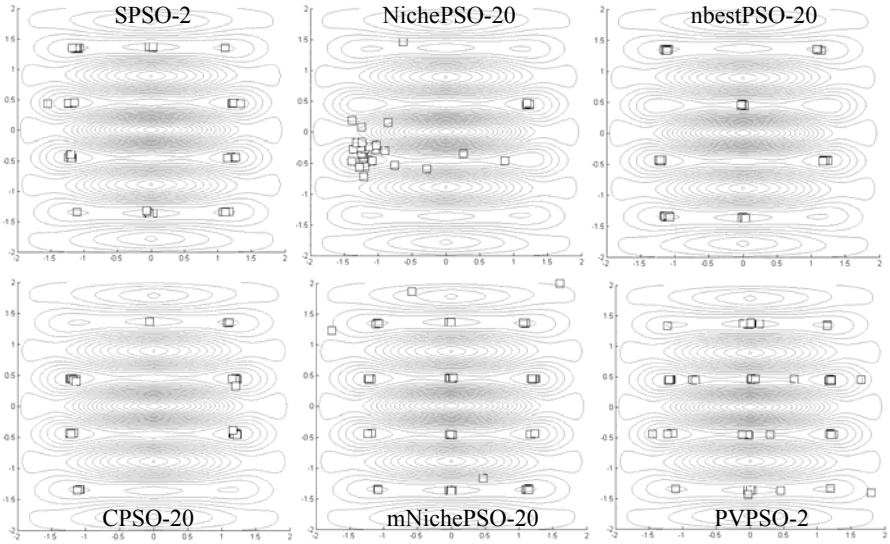

Fig. 3. Particle positions after a sample run of each algorithm on F6

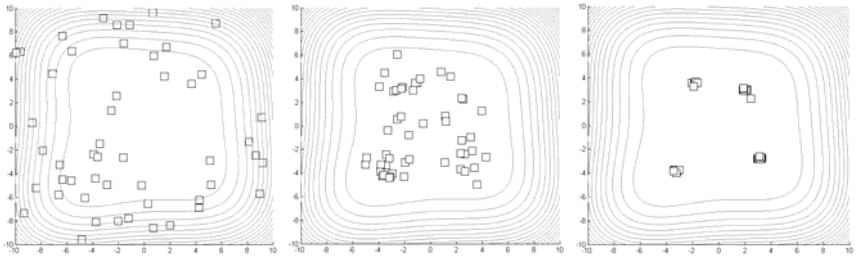

Fig. 4. A sample run of CPSO-20 on F5

5 Conclusions

There are five essential PSO algorithms that have been proposed for multimodal function optimization: SPSO, NichePSO, nbestPSO, UPSO and PVPSO. During the preliminary experiments, it is observed that the performance of NichePSO can be improved significantly restricting the growth of the niche radius beyond a predefined value. mNichePSO performs much better than NichePSO on the benchmark functions, considering the evaluation criteria. In this paper, a new multimodal particle swarm optimization algorithm called CPSO is introduced. CPSO is compared against these approaches. All algorithms introduce additional parameters, requiring fine tuning. Except UPSO, all algorithms are computationally expensive due to the need of

the distance calculations. If the dimensionality increases, the scalability issue might arise, causing this cost to become remarkable. In CPSO, the cost is somewhat reduced by making the computations at a predefined frequency. With its enhanced exploration and exploitation capabilities based on craziness and hill climbing, CPSO has a good performance especially in locating multiple global optima, matching the overall performance of mNichePSO and SPSO on the benchmark functions.

References

1. D. Beasley, D.R. Bull, R.R. Martin, A Sequential Niching Technique for Multimodal Function Optimization, Evolutionary Computation, 1(2), MIT Press, 1993, pp. 101-125.
2. F. van den Bergh, An Analysis of Particle Swarm Optimizers, PhD Thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.
3. F. van den Bergh, A.P. Englebrecht, A study of particle swarm optimization particle trajectories, Information Sciences 176, 2006, pp. 937-971.
4. R. Brits, A.P. Engelbrecht, F. vanden Bergh, Solving Systems of Unconstrained Equations using Particle Swarm Optimization, Int. Conf. on Sys.,Man and Cyber., v.3, 2002, no.pp. 6.
5. R. Brits, A.P. Engelbrecht, F. van den Bergh, A niching particle swarm optimizer, Proc. 4th Asia-Pacific Conf. on Simulated Evolution and Learning, vol. 2, 2002, pp. 692-696.
6. R. C. Eberhart, Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, Proc. of the IEEE Congress on Evolutionary Comp., 2000, pp. 84-88.
7. J. Kennedy, R.C. Eberhart, Particle Swarm Optimization, Proc. IEEE Int. Conf. on N.N., 1995, pp. 1942-1948.
8. J.-P. Li, M.E. Balazs, G.T. Parks, P.J. Clarkson, A Genetic Algorithm using Species Conservation for Multimodal Function Optimization, Journal of Evolutionary Computation, vol. 10, no. 3, 2002, pp. 207-234.
9. X. Li, Adaptively Choosing Neighborhood Bests using Species in a Particle Swarm Optimizer for Multimodal Function Optimization, Proc. of GECCO 2004, LNCS 3102, eds. Deb, K. et al., Springer-Verlag, Seattle, USA, 2004, pp. 105-116.
10. M. Løvbjerg, T.K. Rasmussen, T. Krink, Hybrid Particle Swarm Optimizer with Breeding and Subpopulations, Proc. of the Genetic and Evolutionary Comp. Conf., v. 1, 2001, pp. 469-476.
11. E. Ozcan, C.K. Mohan, Particle Swarm Optimization: Surfing the Waves, Proc. of IEEE Congress on Evolutionary Computation, Piscataway, NJ. 1999, pp. 1939-1944.
12. K. E. Parsopoulos, M. N. Vrahatis, Modification of the particle swarm optimizer for locating all the global minima, Proc. of the ICANNGA, 2001, pp. 324-327.
13. K. E. Parsopoulos, M. N. Vrahatis, UPSO: A Unified Particle Swarm Optimization Scheme, Lecture Series on Comp. and Computational Sci., Vol. 1, Proc. of the Int. Conf. of Computational Methods in Sci. and Eng., 2004, pp. 868-873.
14. I.L. Schoeman, A.P. Engelbrecht, A Parallel Vector-Based Particle Swarm Optimizer, Proc. of the International Conf. on Neural Networks and Genetic Algorithms, 2005, pp. 268-271.
15. D.G. Sotiropoulos, V.P. Plagianakos, M.N. Vrahatis, An evolutionary algorithm for minimizing multimodal functions, Proc. of the Fifth Hellenic- European Conf. on Comp. Math. and its App., vol. 2, 2002, pp. 496-500.
16. F. Streichert, G. Stein, H. Ulmer, A. Zell, A Clustering Based Niching EA for Multimodal Search Spaces, Artificial Evolution, 2003, pp. 293-304.
17. R.K. Ursem, Multinational evolutionary algorithms, Proc. of the 1999 Congress of Evolutionary Computation (CEC-1999), vol. 3, 1999, pp. 1633-1640.