Effect of the dynamic topology on the performance of PSO-2S algorithm for continuous optimization

Abbas El Dor, Patrick Siarry

Effect of the dynamic topology on the performance of PSO-2S algorithm for continuous optimization

Abbas EL DOR[1] and Patrick SIARRY[2]

[1] Ecole des Mines de Nantes, TASC INRIA (CNRS UMR 6241),
4 rue Alfred Kastler, 44300 Nantes, FRANCE
abbas.eldor@mines-nantes.fr
[2] Université de Paris-Est Créteil, LiSSi (E.A. 3956),
122 rue Paul Armangot, 94400 Vitry-sur-Seine, FRANCE
siarry@u-pec.fr

Abstract. PSO-2S is a multi-swarm PSO algorithm using charged particles in a partitioned search space for continuous optimization problems. In order to improve the performance of PSO-2S, this paper proposes a novel variant of this algorithm, called DPSO-2S, which uses the Dcluster neighborhood topologies to organize the communication networks between the particles. Experiments were conducted on a set of classical benchmark functions. The obtained results prove the effectiveness of the proposed algorithm.

1 Introduction

The concept of particle swarm optimization (PSO) is based on social behavior to exchange information between the particles in a swarm. Thus this property can be modelized thanks to a graph: two particles P_i and P_j of the swarm S are connected if a communication can be established between them. The set of edges between each particle P_i and its neighbours Ne_i forms the communication graph, also called the topology. Hence, the chosen topology can greatly affect the performance of the PSO algorithm. In this paper, we present a new dynamic topology, called Dcluster, which is a combination of two existing topologies (Four-clusters [7] and Wheel [6]). This topology was integrated in our proposed algorithm called PSO-2S, introduced in [2]. PSO-2S is a multi-swarm PSO algorithm using charged particles in a partitioned search space for continuous optimization problems. The performance of PSO-2S with the Dcluster topology is analysed and compared to that of PSO-2S without Dcluster, using a set of benchmark test functions. Comparisons show that the use of Dcluster improves significantly the performance of PSO-2S.

2 Particle Swarm Optimization

The particle swarm optimization (PSO) [4] is inspired originally by the social and cognitive behavior existing in the bird flocking. The algorithm is initialized

with a population of particles randomly distributed in the search space, and each particle is assigned a randomized velocity. Each particle represents a potential solution to the problem. The particles fly over the search space, keeping in memory the best solution encountered. At each iteration, each particle adjusts its velocity vector, based on its momentum, influences of its best solution and of the best solution of its neighbors, then computes a new point to be evaluated. The displacement of a particle is influenced by three components:

1. *Physical component*: the particle tends to keep its current direction of displacement;
2. *Cognitive component*: the particle tends to move towards the best site that it has explored until now;
3. *Social component*: the particle tends to rely on the experience of its congeners, then moves towards the best site already explored by its neighbors.

In this paper, the swarm size is denoted by s, and the search space is n-dimensional. In general, a particle i has three attributes: the current position $X_i = (x_{i,1}, x_{i,2}, ..., x_{i,n})$, the current velocity vector $V_i = (v_{i,1}, v_{i,2}, ..., v_{i,n})$ and the past best position $Pbest_i = (p_{i,1}, p_{i,2}, ..., p_{i,n})$. The best position found in the neighborhood of the particle i is denoted by $Gbest_i = (g_{i,1}, g_{i,2}, ..., g_{i,n})$. These attributes are used to update iteratively the state of each particle in the swarm. The objective function to be minimized is denoted by f. The velocity vector V_i of each particle is updated using the best position it visited so far and the overall best position visited by its neighbors. Then, the position of each particle is updated using its updated velocity per iteration. At each step, the velocity of each particle and its new position are updated as follows:

$$v_{i,j}(t+1) = wv_{i,j}(t) + c_1 r_{1_{i,j}}(t)\ [p_{i,j}(t) - x_{i,j}(t)] + c_2 r_{2_{i,j}}(t)\ [g_{i,j}(t) - x_{i,j}(t)] \quad (1)$$

$$x_{i,j}(t+1) = x_{i,j}(t) + v_{i,j}(t+1) \quad (2)$$

$x_{i,j}$ is the position and $v_{i,j}$ is the velocity of the ith particle ($i \in 1, 2, ..., s$) of the jth dimension ($j \in 1, 2, ..., n$). Where w is called inertia weight, c_1, c_2 are the learning factors and r_1, r_2 are two random numbers selected uniformly in the range $[0, 1]$.

3 PSO-2S algorithm

In this section, we present the first version of PSO-2S [2]. PSO-2S is based on three main ideas. The first is to use two kinds of swarms: a main swarm, denoted by S1, and s auxiliary ones, denoted by $S2_i$, where $1 \leq i \leq s$. The second idea is to partition the search space into several zones in which the auxiliary swarms are initialized (the number of zones is equal to the number of auxiliary swarms, thus is equal to s). The last idea is to use the concept of the electrostatic repulsion heuristic to diversify the particles for each auxiliary swarm in each zone.

To construct S1, we propose to perform the auxiliary swarms $S2_i$ several times in different areas, and then each best particle for each $S2_i$ is saved and

considered as a new particle of S1. To do so, the population of each auxiliary swarm is initialized randomly in different zones (each $S2_i$ is initialized in its corresponding zone i). After each of these initializations, K displacements of particles, of each $S2_i$, are performed in the same way of standard PSO. Then the best solution found by each auxiliary swarm is added to S1. The number of initializations of $S2_i$ is equal to the number of particles in S1.

As we mentioned above the second idea is to partition the search space $[min_d,\ max_d]^N$ into several zones (max_{zone} zones). Then, we calculate the $center_d$ and the $step_d$ of each dimension separately, according to (3) and (4). The $step_d$ are similar in the case of using a square search space.

$$center_d = (max_d - min_d)/2 \tag{3}$$

$$step_d = center_d/max_{zone} \tag{4}$$

where max_{zone} is a fixed value, and d is the current dimension ($1 \leq d \leq N$).

The sizes of the zones of the partitioned search space are different ($Z_1 < Z_2 < \ldots < Z_{max_{zone}}$). Therefore, the number of particles in $S2_i$, denoted by $S2_{isize}$, depends on its corresponding zone size. Indeed, a small zone takes less particles and the number of particles increases when the zone becomes larger. The size of each auxiliary swarm is calculated as follows:

$$S2_{isize} = num_{zone} * nb_{particle} \tag{5}$$

where num_{zone} = 1, 2, ..., max_{zone}, is the current zone number and $nb_{particle}$ is a fixed value. After the initializations of the auxiliary swarms in different zones (Z_i, $S2_i$), an electrostatic repulsion heuristic is applied to diversify the particles and to widely cover the search space [3]. This technique is used in an agent-based optimization algorithm for dynamic environments [5]. Therefore, this procedure is applied in each zone separately, hence each particle is considered as an electron. Then a force of $1/r^2$ is applied, on the particles of each zone, until the maximum displacement of a particle during an iteration becomes lower than a given threshold ϵ (where r is the distance between two particles, ϵ is typically equal to 10^{-4}). At each iteration of this procedure, the particles are projected in the middle of the current zone, before reapplying the heuristic repulsion.

4 Dynamic cluster topology (Dcluster)

Dcluster is a dynamic topology that works as follows [1]. At each iteration, the particles are sorted in a list according to their personal best fitness in increasing order, so that the worst particle has an index equal to 1 in the list (the size of the list is equal to the size of the swarm). Then, the list is partitioned into several sub-lists which correspond to a cluster in the proposed topology. The first cluster which has the "worst" particle, called central cluster, is placed in the center of the topology. Each particle of the central cluster is connected to other clusters by one of their particles; the first worst particle of the central cluster is linked

to the worst particle in the second cluster, the second one is linked to the third cluster also by its worst particle, and so on, as in Figure 1-(a). All clusters in this topology, including the central cluster, have a fully connected neighborhood. The reason why the central cluster is linked to each other cluster by only one gateway and with the worst particle of the latter is to avoid a premature convergence to a local optimum by slowing down the propagation of the information in the whole swarm. Figure 1-(b) illustrates the final structure of the proposed topology.

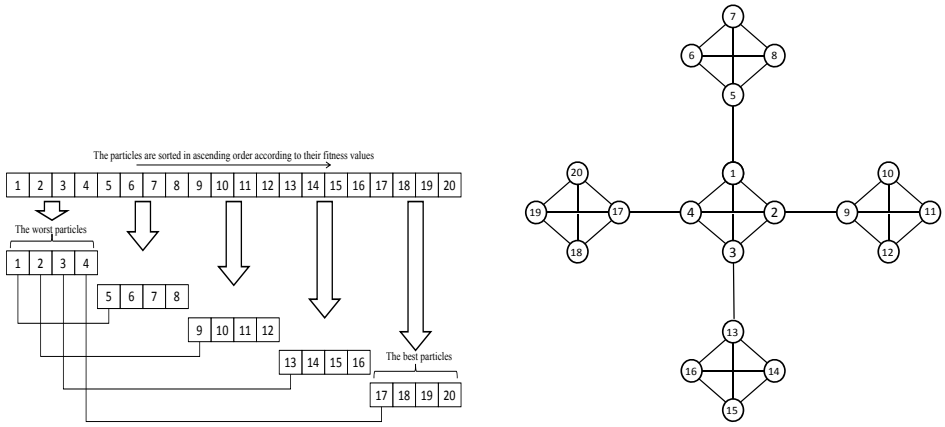

Fig. 1. (a) - The partitioning of the list into sub-lists. (b) - The structure of Dcluster topology.

5 Experimental results

Table 1 presents the settings of each problem, the number of function evaluations (Max. FEs), the success rate and the mean best value of 100 runs. The best results among those obtained by the two algorithms are shown in bold. From the experiments, we can notice that DPSO-2S (PSO-2S using Dcluster topology) obtains the best results on most of the functions used. Hence, DPSO-2S algorithm outperforms PSO-2S, except for *Rosenbrock* and *Shifted Rosenbrock*. Thus, this algorithm leads to a significant improvement over the previous PSO-2S.

6 Conclusion

In this paper, a new dynamic topology, called Dcluster, based on two static neighbourhood topologies was presented. Dcluster was integrated to our multi-swarm algorithm PSO-2S. Experimental results indicate that Dcluster improves the search performance of the previous algorithm.

Function	Search space	Acceptable error	Max. FEs	DPSO-2S		PSO-2S	
				Mean best	*Suc. rate*	*Mean best*	*Suc. rate*
Rosenbrock	$[-10, 10]^{30}$	0.0001	40000	2.50e+001	0.0%	**2.28e+001**	0.0%
Ackley	$[-32, 32]^{30}$	0.0001	40000	**9.40e-003**	**99%**	3.54e-001	69%
Griewank	$[-100, 100]^{30}$	0.0001	40000	**2.19e-003**	**78%**	3,88e-003	72%
Rastrigin	$[-10, 10]^{30}$	0.0001	40000	**1.34e+000**	**30%**	2.16e+000	25%
Sh. Rosenbrock	$[-100, 100]^{10}$	0.01	100000	5.25e+000	5%	**5.98e-001**	**75%**
Sh. Ackley	$[-32, 32]^{30}$	0.0001	100000	**6.26e-002**	**95%**	1.88e-001	86%
Sh. Griewank	$[-600, 600]^{30}$	0.0001	100000	**5.16e-003**	**66%**	6.11e-003	61%
Sh. Rastrigin	$[-5, 5]^{30}$	0.0001	100000	**3.14e+001**	**0,0%**	5,36e+001	0.0%

Table 1. Results of DPSO-2S using Dcluster topology and PSO-2S.

In conclusion, the improvement of the PSO-2S algorithm due to the integration of Dcluster topology opens the gate to apply this dynamic topology in other PSO algorithms.

References

1. El Dor A., Lemoine D., Clerc M., P. Siarry, L. Deroussi, and M. Gourgand. Dynamic Cluster in Particle Swarm Optimization algorithm. *Natural Computing*, October 2014. DOI: 10.1007/s11047-014-9465-2.
2. El Dor A., Clerc M., and Siarry P. A multi-swarm PSO using charged particles in a partitioned search space for continuous optimization. *Computational Optimization and Applications*, 53:271–295, 2012.
3. Conway J. and Sloane N. Sphere packings, lattices and groups. *Springer (3rd ed.)*, 1999.
4. Kennedy J. A new optimizer using particle swarm theory. *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43, 1995.
5. Lepagnot J., Nakib A., Oulhadj H., and Siarry P. A new multiagent algorithm for dynamic continuous optimization. *International Journal of Applied Metaheuristic Computing*, 1(1):16–38, 2010.
6. J. Kennedy and R. Mendes. Population structure and particle swarm performance. In *Proceedings of the 2002 IEEE Congress on Evolutionary Computation, CEC'02*, pages 1671–1676, Honolulu, HI, USA, 2002.
7. R. Mendes, J. Kennedy, and J. Neves. The Fully Informed Particle Swarm: Simpler, Maybe Better. *IEEE Trans. Evolutionary Computation*, 8(3):204–210, June 2004.