Hierarchical Dynamic Neighborhood Based Particle Swarm Optimization for Global Optimization

Pradipta Ghosh[1], Hamim Zafar[1], Swagatam Das[1] and Ajith Abraham [2, 3]

[1]Department of Electronics and Telecommunication Engineering, Jadavpur University, Kolkata 700 032, India
[2]Faculty of Computer Science and Electrical Engineering, VSB – Technical University of Ostrava, Czech Republic
[3]Machine Intelligence Research Labs (MIR Labs), Seattle, WA, USA
ajith.abraham@ieee.org

Abstract— **Particle Swarm Optimization (PSO) is arguably one of the most popular nature-inspired algorithms for real parameter optimization at present. In this article, we introduce a new variant of PSO referred to as Hierarchical D-LPSO (Dynamic Local Neighborhood based Particle Swarm Optimization). In this new variant of PSO the particles are arranged following a dynamic hierarchy. Within each hierarchy the particles search for better solution using dynamically varying sub-swarms i.e. these sub-swarms are regrouped frequently and information is exchanged among them. Whether a particle will move up or down the hierarchy depends on the quality of its so-far best-found result. The swarm is largely influenced by the good particles that move up in the hierarchy. The performance of Hierarchical D-LPSO is tested on the set of 25 numerical benchmark functions taken from the competition and special session on real parameter optimization held under IEEE Congress on Evolutionary Computation (CEC) 2005. The results have been compared to those obtained with a few best-known variants of PSO as well as a few significant existing evolutionary algorithms.**

Keywords - PSO, local PSO,hierarchy, D-LPSO, Hierarchical D-LPSO

I. INTRODUCTION

The concept of particle swarms originated from the simulation of the social behavior commonly observed in animal kingdom and evolved into a very simple but efficient technique for global numerical optimization in recent past. The Particle Swarm Optimization (PSO) [1, 2] as it is called now, does not require any gradient information of the function to be optimized, uses only primitive mathematical operators, and is conceptually very simple. Since its inception in 1995, PSO has attracted a great deal of attention of the researchers all over the globe resulting into several variants of the basic algorithm, theoretical and empirical investigations of the dynamics of the particles, parameter selection and control, and applications of the algorithm to a wide spectrum of real world problems from diverse fields of science and engineering. For Evolutionary Algorithms (EAs) search for an optimum is an iterative process that depends on some random decisions. For a comprehensive knowledge on the foundations, perspectives, applications of PSO see [1−2] [4−6]. The effectiveness of PSO is mainly attributed to the efficient communication of information among the search agents. PSO has already been applied to numerous benchmark as well as real world optimization problems successfully.

PSO exploits the dynamics of a population of trial solutions or search-agents that collaborate for finding better solutions. PSO combines *cognition only model* that values only the self-experience and *social only model* that takes into account the experience of neighbors. The algorithm uses a set of particles astrogating over a search space and moving towards a promising position to locate a global optimum. Each particle stands for a potential solution to an optimization problem. Initially the particles are distributed randomly over the search space each one endowed with a random velocity, and the goal is to converge to the global optimum of a function. During their journey with discrete time iterations, the velocity of each particle in the next iteration is determined by the best position found by the particles of the swarm (*gbest* as the *social* component), the best personal position of the particle (*pbest* as the *cognitive* component), and its current velocity (the *memory* term).

Being a stochastic search process PSO is not free from false and/or premature convergence, especially over multimodal fitness landscapes. As there is a direct link of the information flow between particles and *gbest,* multifariousness is lost. As a result probability of being trapped in local optima increases that result in premature convergence. Various modifications and PSO variants have been proposed to eradicate this problem. The modifications can be regarded as algorithmic components that provide an improved performance. These algorithmic components may be integrated in the particles' velocity update rule (1) or as stand-alone algorithms that are used as components of hybrid PSO algorithms.

In this article, we propose a new variant of PSO called Hierarchical D-LPSO. In Hierarchical D-LPSO, a particle is influenced by its own so far best position and by the best position of the particle that is directly above it in the hierarchy. All particles are arranged in a tree that forms the hierarchy so that each node of the tree contains exactly one particle. Particles can move up and down the hierarchy depending on the solution it has obtained. Within each level of hierarchy the particles search for better solution using a dynamically varying neighborhood. Each hierarchy involves a no of sub-swarms of particles that search for better solution and information is exchanged between the sub-swarms

978-1-4244-7833-0/11/$26.00 ©2011 IEEE

resulting in increased diversity. Performance of the proposed PSO is compared with other those of Unified PSO (UPSO) [4], Comprehensive Learning PSO (CLPSO) [5], wFIPS [6], Dynamic Multi-Swarm PSO (DMS-PSO) [7], Cooperative PSO (CPSO) [5], Differential Evolution (DE) [8] and one of its significant adaptive variants [9].

The rest of the paper is arranged in the following way: Section II contains the an overview of PSO algorithm, Section III gives a brief overview of L-PSO, Section IV describes Hierarchical PSO, Section V presents D-LPSO algorithm and Section VI describes our Novel method abbreviated as Hierarchical D-LPSO, Section VII contains a comparative study of the algorithms over CEC (Congress on Evolutionary Computation) 2005 benchmark problems [10] and Section VIII concludes this paper..

II. An Overview of PSO Algorithm

The classical PSO starts with the random initialization of a population of candidate solutions (particles) over the fitness landscape. However, unlike other evolutionary computing techniques, PSO uses no direct recombination of genetic material between individuals during the search. Rather it works depending on the social behavior of the particles in the swarm. Therefore, it finds the global best solution by simply adjusting the trajectory of each individual towards its own best position and toward the best particle of the entire swarm at each time-step (generation). In a *D*-dimensional search space, the position vector of the *i*-th particle is given by $\vec{X}_i = (x_i^1, x_i^2, \cdots, x_i^D)$ and velocity of the *i*-th particle is given by $\vec{V}_i = (v_i^1, v_i^2, \cdots, v_i^D)$. Positions and velocities are adjusted and the objective function to be optimized $f(\vec{X}_i)$ is evaluated with the new coordinates at each time-step. The velocity and position update equations for the *d*-th dimension of the *i*-th particle in the swarm may be represented as:

$$v_i^d = \omega * v_i^d + c_1 * rand1_i^d * (pbest_i^d - x_i^d) + c_2 * rand2_i^d * (gbest^d - x_i^d),$$
$$x_i^d = x_i^{d-1} + v_i^d, \quad (1)$$

where c_1 and c_2 are the acceleration constants, c_1 controls the effect of the personal best position, c_2 determines the effect of the best position found so far by any of the particles, $rand1_i^d$ and $rand2_i^d$ are two uniformly distributed random numbers in the range [0, 1]. ω is the inertia weight that balances between the global and local search abilities and takes care of the influence of the previous velocity vector. $pbest_i = (pbest_i^1, pbest_i^2, \cdots, pbest_i^D)$ is the best previous position yielding the best fitness value $pbest_i$ for the i^{th} particle and $gbest = (gbest^1, gbest^2, \cdots, gbest^D)$ is the best position discovered by the entire swarm.

III. Local Neighborhood Based PSO

The two main variants are global_best (or *gbest*) PSO and local_best (*lbest*) PSO. Some other variants limit the velocity of a particle by a maximal value $V_{\max}$, some variant linearly varies ω. In *lbest* PSO, each particle's velocity is modified according to its personal best and the best performance achieved so far within its neighborhood instead of learning from the personal best and the best position achieved so far by the whole population in the global version. The velocity updating equation becomes:

$$v_i^d = \omega * v_i^d + c_1 * rand1_i^d * (pbest_i^d - x_i^d) + c_2 * rand2_i^d * (lbest_i^d - x_i^d) \quad (2)$$

where $lbest_i = (lbest_i^1, lbest_i^2, \cdots, lbest_i^D)$ is the best position achieved within its neighborhood. To increase the diversity among the particles of a swarm various mechanisms has been designed. The topology of the neighborhood plays a substantial role in PSO, and different neighborhood topologies have been investigated for PSO. In the *lbest* model of PSO, the neighborhood is defined by a ring topology based on the particles' index. For the improvement of this *lbest* model of PSO, different neighborhood structures are proposed and discussed in literature. There are some variants, which use multi-swarm [7], subpopulation [7]. Sub-groups may be treated as special neighborhood structures. In the existing local versions of PSO with different neighborhood structures and the multi-swarm PSOs, the swarms are predefined or dynamically adjusted according to the distance. The dynamic multi-swarm optimizer uses a dynamic or randomly assigned topology. We use the dynamic topology i.e. dynamically varying sub populations along with the hierarchical version of the PSO and obtain an improved variant of PSO which shows good performance on the 30-*D* test functions obtained from the competition and special session on real-parameter optimization held under CEC 2005.

IV. A Brief Overview of Hierarchical PSO

In the hierarchical version of PSO the particles are arranged in a hierarchy and it defines the neighborhood structure. Each particle is neighbored to itself and also its parent in the hierarchy. The regular treelike hierarchies are followed. The hierarchy is defined by the *height,* the *branching degree* [11], i.e., the maximum number of children of the inner nodes, and the *total number of nodes* of the corresponding tree. In this hierarchy all inner nodes have the same number of children, but the inner nodes on the deepest level might have a smaller number of children. As a result the maximum difference between the numbers of children of inner nodes on the deepest level is at most one. The best particles of the swarm become highly influential by the upward and downward movement of the particles in the hierarchy. In every iteration, the new positions of the particles are determined between the evaluation of the objective function and velocity update. The best solution of j^{th} particle in a node of the tree is compared

to the best solution obtained by the particles in the child nodes. This is done for every particle in that node. If best solution obtained by any particle (say i^{th} particle) in the child node is better than that of j^{th} particle then the particles i and j swap their places. These comparisons start from the top of the hierarchy and then proceed in a breadth-first manner down the tree. The particle having global best position of the hierarchy moves up one level of the hierarchy at every iteration. The velocity of a particle is modulated by its own so far best position and by the best position of the individual that is directly above in the hierarchy.

In case of H-PSO [11] neighborhood of a particle changes constantly depending on the fitness development of the individuals. The changing arrangement of the particles can help preserving diversity in the search. The arrangement of the particles leads to a different influence for the particles at different positions. The particle with the currently best found solution can (indirectly) influence all the other particles after it has reached the top of the hierarchy.

The structure of the hierarchy, the branching degree *d* influences the optimization behavior of H-PSO. For example, if branching degree is higher, then performance might be better initially, on the other hand due to a smaller value of *d* performance of finding best solution may be worse in the beginning of the optimization process but it might improve the objective function value further in the end of the optimization process. For this reason the branching degree is changed dynamically. When the branching degree is decreased from d to d-1, the hierarchy is traversed starting at the root node. This is done so that always one of the direct sub trees below the considered node is removed, if the number of children exceeds the new required branching degree. The removal of sub tree is based on the quality of the particles in the topmost nodes of all sub trees of the considered nodes, i.e., all children of the considered node. This procedure is repeated for the entire tree. After this removal of sub tree the remaining tree has branching degree *d*-1 and fewer nodes than before. The removed nodes are then evenly inserted at the bottom of the hierarchy. The removed nodes are appended one by one so that the number of children of all nodes on the second last level differs by at most one. If all of these nodes have *d*-1 children, a new level is added to the hierarchy and the procedure is continued until all removed nodes are reinserted. The branching degree reduction is done in every f_{adapt} th iteration, this f_{adapt} is called decrease frequency. Branching degree is decreased by k_{adapt} known as decrease step size. For $k_{adapt} > 1$ the reduction procedure is applied consecutively (i.e., the branching degree is always reduced in steps of 1) until the hierarchy has the required branching degree. This is done until a certain minimum branching degree is reached. To choose which *sub tree* are to be removed two strategies are used, removing the *sub tree* with the worst root node or the one with the best root node.

V. PROPOSED D-LPSO ALGORITHM

The Dynamic Local Neighborhood based Particle Swarm Optimization (D-LPSO) is a variant of PSO constructed based on the local version of PSO employing a new neighborhood topology. In case of PSO it has been found that satisfactory results can be obtained using smaller population size. PSO with smaller neighborhoods has better performance on complex problems also. In case of D-LPSO smaller neighborhoods are used. As a result the convergence velocity of the population decreases, diversity increases and better solutions are achieved for multi-modal problems.

In order to slow down the population's convergence velocity and increase diversity and achieve better results on multimodal problems, in the D-LPSO, small neighborhoods are used. The population is divided into small sized swarms. Each sub-swarm uses its own members to search for better area in the search space. Since the small sized swarms are searching using their own best historical information, they are easy to converge to a local optimum because of PSO's convergence property. In order to avoid it we must allow information exchange among the swarms. And in the information exchange schedule, we want to keep more information including the good ones and the not so good ones to add the varieties of the particles and achieve larger diversity. So a randomized regrouping schedule is introduced to make the particles have a dynamic changing neighborhood structures. Each sub-swarm containing at most three particles search for better location and they may converge to near a local optimum. After regrouping, the particles previously belonging to a common sub-swarm now belong to different sub-swarms and get the opportunity to modify their velocity and position learning from the new swarm members. During regrouping the particles are distributed among different swarms randomly. In every generation, the population is regrouped randomly and starts searching using a new configuration of small swarms. In this way, the information obtained by each swarm is exchanged among the swarms as a particle belongs to different swarms during the search process and it carries the information obtained in the previous swarm and uses this information to influence other particles' movement in the new swarm. With the randomly regrouping schedule, particles from different swarms are grouped in a new configuration so that each small swarm's search space is enlarged and better solutions are possible to be found by the new small swarms.

VI. HIERARCHICAL D-PSO ALGORITHM

In this algorithm Hierarchical PSO is combined with D-LPSO. The steps of the proposed PSO algorithm is as follows:

Step1. The Initialized population is divided into stages. First stage contains 1 particle; next stage contains maximum *n* elements (Initially *n* = 2). *N*-th stage can contain maximum n^N particles.

Step2. Each particle is assigned randomly a parent from the previous stage except the particle on 1st stage

Step3. Evaluate each particles Fitness

Step4. Now D-LPSO is applied along each stage except 1st stage.

Step5. Now each particle compares its fitness with its assigned parent. If its position is better than its parent's position, they are swapped.

Step6. Step2 to Step5 continues until next $1/5^{th}$ of the total FEs is completed. Now $n = n+1$, and Step1 takes place again.

Step7. When maximum no of FEs is reached all the processes are stopped and the result is shown.

In the velocity update equation we have used a constriction factor to avoid the unlimited growth of the particles' velocity. This was proposed by Clerc and Kennedy [12]. Equation 2 becomes

$$V_i^d = \chi * (\omega * V_i^d + c_1 * rand1_i^d * (pbest_i^d - X_i^d) + c_2 * rand2_i^d * (lbest_i^d - X_i^d)) \quad (3)$$

χ is the constriction factor given by

$$\chi = 2 / \left| 2 - c - \sqrt{c^2 - 4c} \right| \quad (4)$$

$$c = \sum_i c_i$$

This procedure is shown in Figure 1.

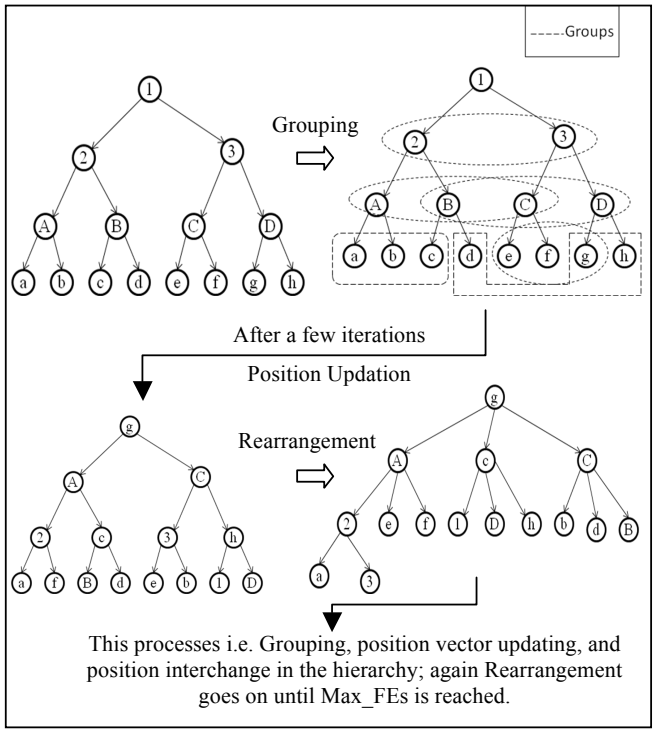

Figure 1: Hierarchical D-LPSO's Search Process

The pseudo codes of Step1 to Step 5 are given below. Step 6 and Step 7 can be easily coded.

Step1

N=total no of particles.
FEs= No of FEs covered;
Max_FEs=Max No of FEs
Count =Max no of particles in second stage.
Icount= Max no of particles in any stage *Stage(1,1)*= Any particle from the population.
Stagecount=2;
Icount=Count;
While 1
    For *i*=1: *Icount*
        Stage (Stagecount, i) =any particle from the rest of the population
        If All the particles Covered
            Break;
        End
    End
    If all the particles are covered
        Break;
    End
    *Icount=Icount*Count*;
    Stagecount=Stagecount+1;
End

Step2

(Stage (1,1))= Stage (1,1);
Icount=Count;
For *j*=1: *Stagecount*
    For *i*=1: *Icount*
        Parent (Stage (Stagecount, i)) =any particle from the previous stage.
    End
        *Icount=Icount*Count*;
End

Step3

For *i*=1:*N*
    Evaluate Each particle's Fitness
End

Step4

For *j*=1: *Stagecount*
    Apply D-LPSO;
End

Step5

For *j*=1: *Stagecount*
    For *i*=1: *Icount*
        If *Fitness (Parent (Stage (Stagecount, i)))* <*Fitness (Stage (Stagecount, i))*
            Swap the particles.
        End
    End
        *Icount=Icount*Count*;
End

In the above algorithm we are repeating steps 2 to 5 for $1/5^{th}$ of the total FEs and after that we start the process again. It has been found empirically that good results are obtained by restarting the process after $1/5^{th}$ of the total FEs. Steps 2 to 5 yield sufficiently good result within this no of FEs and more no. of FEs are not required. When applying D-LPSO in step 4

we create sub-swarms containing at most 3 particles within each level of hierarchy, the no of sub-swarms in a level of hierarchy depends on the no of particles in that particular level.

VII. Experimental Results

A. Benchmark Functions Used:

For the evaluation of the performance of the new variant of PSO a test-bed of twenty-five well - known boundary-constrained benchmark functions has been used. These functions constituted the benchmark of CEC-2005 competition on single objective optimization. These functions can be divided into two groups as follows

- Unimodal Functions: f_1 to f_5
- Multimodal Functions: f_6 to f_{25}

Among these functions seven are simple test functions, two others are expanded functions (Whitley et al. 1996).The remaining eleven functions are hybrid composition functions. Only f_1 and f_9 are separable. These functions were designed to test an optimizer's ability to locate a global optimum under a variety of circumstances:

- Function landscape is highly conditioned
- Function landscape is translated
- Function landscape is rotated
- Optimum lies in a narrow basin
- Optimum lies on a bound
- Optimum lies beyond the initial bounds
- Function is not continuous everywhere
- Gaussian N(0,1) noise is to the function evaluation
- Bias is added to the function evaluation

The detailed information about the test functions is available on–line at:
http://www.ntu.edu.sg/home/EPNSugan

B. Algorithms Compared:

The results of Hierarchical D-LPSO on the above test bed have been compared to the following algorithms:

- DE/rand/1/bin [8]
- jDE with NP=100,τ1=τ2=0.1 [9]
- CLPSO [5]
- UPSO [4]
- DMS-PSO [7]
- wFIPS [6]
- CPSO [5]

Among the above seven algorithms the first one is a classical DE, the next algorithm is a DE variants, the next five algorithms are various PSO variants. The results of the compared algorithms have been obtained from 25 independent runs on each of twenty-five numerical benchmarks.

C. Simulation Strategies:

Functions f_1 to f_{25} were tested in 30-dimensions (30D). The following specifications are used. Each run for all the algorithms were terminated when the number of Function Evaluations (FEs) exceed 3e+05. Parametric set-up for all the benchmark problems considered here: c_1=2.05, c_2=2.05, w=0.793, χ=0.729. Our algorithm is tested on a Pentium core 2 duo machine with 1 GB RAM and 2.00 GHz speed.

D. Results on Benchmark Functions:

The results of Hierarchical D-LPSO, DE/rand/1/bin, jDE, CLPSO, UPSO, DMSPSO, wFIPS, and CPSO have been shown below in Tables 1 to 5. The results are presented in terms of mean and standard deviations obtained from 25 independent runs on each of twenty-five numerical benchmarks for 30 Dimensions. In order to determine the statistical significance of the advantage of the hierarchical D-LPSO over other algorithms, a non-parametric statistical test, called Wilcoxon's ranksum test [13, 14] is applied on the mean error found at the 5% significance level and the results are shown in Table 6. The numerical values 1, 0, -1 represent that other methods are statistically superior to, equal to or inferior to the proposed algorithm.

Table 1: Mean and std.(in parentheses) of error values for functions 1-5

Algorithms	f_1	f_2	f_3	f_4	f_5
	Mean (Std)	Mean (Std)	Mean (Std)	Mean (Std)	Mean (Std)
DE/rand/1/bin	1.3554e-29 (1.3893e-15)	5.4912e-08 (1.2449e-07)	2.8911e+05 (1.9321e+05)	5.0455e-01 (8.5812e-01)	**2.3500e+02 (1.8312e+02)**
jDE	1.0000e-29 (5.3453e-16)	7.5064e-06 (7.3804e-06)	2.2663e+05 (1.6085e+05)	**2.7305e-01 (1.5490e-01)**	1.1108e+03 (3.7238e+02)
UPSO	1.5773e-28 (1.2721e-17)	2.4150e+01 (9.0551e-01)	3.4662e+03 (5.3453e-16)	2.1533e+03 (5.1222e+00)	1.4190e+03 (1.2135e+01)
CLPSO	6.9032e-26 (4.4361e-14)	3.9386e+05 (6.8032e+00)	7.4046e+07 (2.5412e+05)	1.4314e+04 (1.0954e+01)	1.2266e+04 (8.2250e+03)
DMS-PSO	3.3149e-29 (2.431e-10)	0.8968e+00 (7.9441e+00)	7.9812e+06 (2.0285e+02)	7.9277e+02 (1.1354e+01)	7.5656e+03 (1.3555e+01)
wFIPS	2.3896e-015 (1.4035e-07)	2.8500e+03 (4.6839e+02)	2.4097e+03 (7.8151e+05)	1.4268e+03 (4.1457e+01)	1.8613e+03 (3.8819e+03)
CPSO	2.8547e-09 (1.0947e-05)	1.1777e+03 (3.7518e+01)	1.4116e+03 (5.1021e+04)	3.2427e+04 (2.1253e+01)	1.2955e+04 (1.7723e+03)
Hierarchical D-LPSO	**1.0000e-30 (9.3837e-20)**	**1.3939e-22 (7.8076e-03)**	**1.0226e+03 (2.3400e+01)**	8.5560e+01 (2.5674e+01)	3.3301e+03 (9.8939e+00)

Table 2: Mean and std.(in parentheses) of error values for Functions 6-10

Algorithms	f_6	f_7	f_8	f_9	f_{10}
	Mean (Std)	Mean (Std)	Mean (Std)	Mean (Std)	Mean (Std)
DE/rand/1/bin	3.7711e+00 (2.7176e+00)	9.6555e-01 (9.1416e-02)	2.0908e+01 (6.2577e-02)	5.6843e-14 (1.0000e-15)	6.1665e+01 (4.5634e+01)
jDE	1.1196e+01 (1.3987e+00)	9.8597e-03 (3.4824e-03)	2.0955e+01 (2.5067e-02)	**1.2737e-15 (1.0000e-15)**	5.2547e+01 (4.4660e+00)
UPSO	1.4676e+01 (6.9411e+01)	7.4781e-03 (5.1123e-01)	2.1002e+01 (5.1451e-02)	8.0541e+01 (1.1453e+01)	1.1709e+02 (2.2452e+01)
CLPSO	2.7570e+01 (3.0942e+01)	1.4496e-01 (1.6216e-01)	2.1431e+01 (3.1455e-01)	1.4223e+01 (1.9204e+01)	1.7357e+02 (5.7628e+01)
DMS-PSO	5.0691e+01 (4.2186e+01)	0.0148e+00 (8.1534e-01)	2.1009e+01 (4.7572e-01)	2.1889e+01 (1.6371e+01)	1.6018e+02 (2.6161e+01)
wFIPS	2.7808e+01 (7.4091e+00)	0.0224e+00 (5.4664e-01)	2.0893e+01 (7.8543e-01)	8.3434e+01 (1.9281e+01)	1.9873e+02 (5.4815e+01)
CPSO	1.2233e+02 (2.5279e+01)	4.6963e-01 (1.9951e+00)	2.0431e+01 (9.1522e-01)	3.9923e-09 (1.8421e+01)	4.7458e+02 (3.0921e+01)
Hierarchical D-LPSO	**5.7921e-01 (2.0167e+00)**	**3.1086e-15 (9.7520e-04)**	**2.0000e+01 (5.1185e-04)**	6.9093e+00 (2.9427e+00)	**3.2758e+01 (2.9123e+00)**

Table 3: Mean and std.(in parentheses) of error values for Functions 11-15

Algorithms	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
	Mean (Std)	Mean (Std)	Mean (Std)	Mean (Std)	Mean (Std)
DE/rand/1/bin	3.2611e+01 (1.0990e+01)	8.4356e+03 (6.2276e+03)	4.5166e+00 (2.2655e+00)	1.3378e+01 (3.4756e-01)	4.8470e+02 (2.1460e+01)
jDE	3.1370e+01 (2.3952e+00)	3.8376e+04 (6.5374e+03)	1.6568e+00 (9.0313e-01)	1.3545e+01 (9.9402e-02)	2.9642e+02 (1.8711e+01)
UPSO	2.9326e+01 (1.3453e+00)	2.6078e+04 (1.9759e+03)	8.3413e+00 (5.9199e+00)	1.3823e+02 (1.2345e+00)	4.0485e+02 (3.9841e+01)
CLPSO	2.7570e+01 (8.7291e+00)	1.4496e+03 (1.2144e+02)	1.8319e+00 (3.5519e+00)	1.4223e+01 (1.8566e-01)	2.7357e+02 (1.8805e+01)
DMS-PSO	3.763e+01 (3.9221e+00)	1.8494e+05 (2.3081e+03)	3.9243e+00 (1.0025e+00)	1.3498e+01 (2.3052e+00)	2.4561e+02 (2.1109e+01)
wFIPS	3.9810e+01 (7.2393e+00)	9.9352e+05 (8.7152e+02)	1.4137e+01 (1.0723e+01)	1.4137e+01 (5.6714e+00)	3.2362e+02 (7.3117e+01)
CPSO	4.0380e+01 (5.9065e+00)	3.1445e+03 (8.9612e+02)	1.6102e+00 (1.0042e+00)	1.3852e+01 (1.9023e-01)	7.3132e+02 (2.7126e+01)
Hierarchical D-LPSO	**2.4588e+01 (7.7643e-01)**	**2.2635e+02 (1.0456e+02)**	**1.4646e+00 (4.8823e-01)**	**1.1955e+01 (1.9311e-01)**	**2.1324e+02 (1.2044e+01)**

Table 4: Mean and std.(in parentheses) of error values for Functions 16-20

Algorithms	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}
	Mean (Std)	Mean (Std)	Mean (Std)	Mean (Std)	Mean (Std)
DE/rand/1/bin	2.8231e+02 (1.1811e+01)	3.0931e+02 (1.5800e+01)	9.1311e+02 (8.4333e-01)	9.1394e+02 (1.2112e+00)	9.1345e+02 (1.1643e+00)
jDE	1.2854e+02 (4.0730e+01)	1.6189e+02 (4.7251e+01)	8.6111e+02 (1.8705e+00)	8.4801e+02 (3.1790e+00)	8.5466e+02 (9.5496e-01)
UPSO	1.9239e+02 (1.3453e+01)	2.3493e+02 (1.5553e+01)	8.5899e+02 (8.5349e+00)	8.3672e+02 (9.9335e+00)	8.3458e+02 (5.3453e+00)
CLPSO	1.6647e+02 (1.1472e+02)	1.6647e+02 (1.8328e+01)	9.1426e+02 (4.3234e+00)	9.1190e+02 (2.3876e+00)	9.1460e+02 (2.8968e+00)
DMS-PSO	1.5063e+02 (4.1217e+01)	1.2051e+02 (1.7493e+01)	8.4715e+02 (7.1184e+00)	8.4156e+02 (1.2033e+01)	8.3903e+02 (6.0242e+00)
wFIPS	2.5374e+02 (2.6231e+01)	2.4396e+02 (3.7383e+01)	8.5259e+02 (1.9271e+01)	8.5058e+02 (2.8134e+01)	8.4261e+02 (1.5721e+01)
CPSO	2.7883e+02 (7.0912e+01)	4.4672e+02 (3.8973e+01)	9.1759e+02 (1.3845e+01)	9.5649e+02 (3.3420e+00)	9.6774e+02 (1.5644e+00)
Hierarchical D-LPSO	**8.9347e+01 (1.4668e+00)**	**8.4512e+01 (9.8907e+00)**	**8.2927e+02 (8.7532e+00)**	**8.3044e+02 (1.2234e+00)**	**8.3074e+02 (4.6854e-01)**

Table 5: Mean and std.(in parentheses) of error values for Function s21-25

Algorithms	f_{21}	f_{22}	f_{23}	f_{24}	f_{25}
	Mean (Std)	Mean (Std)	Mean (Std)	Mean (Std)	Mean (Std)
DE/rand/1/bin	5.8188e+02 (2.6267e+01)	9.6457e+02 (1.1433e+01)	6.2123e+02 (3.0612e+01)	3.1411e+002 (3.2262e+01)	7.8612e+02 (2.1746e+01)
jDE	8.6002e+02 (1.1361e+00)	5.0340e+02 (2.9115e+00)	6.1835e+02 (4.5481e+00)	2.1081e+02 (2.8842e+00)	2.1153e+02 (1.3637e+00)
UPSO	8.7297e+02 (5.2457e+01)	7.5716e+02 (1.6531e+01)	8.8250e+02 (6.7905e+01)	2.2900e+02 (3.5557e+01)	2.2523e+02 (7.8321e+00)
CLPSO	5.1123e+02 (9.5745e+01)	9.5770e+02 (7.9452e+00)	5.3616e+02 (7.7860e+00)	2.1000e+02 (3.8643e-04)	2.1100e+02 (9.1113e+01)
DMS-PSO	8.6685e+02 (3.4713e+01)	5.2356e+02 (1.2063e+01)	8.7736e+002 (2.8510e+01)	2.1804e+02 (1.0258e+01)	2.2455e+02 (1.9082e+00)
wFIPS	8.6328e+02 (7.1084e+01)	5.2168e+02 (2.3153e+01)	8.6622e+02 (2.1347e+01)	2.1798e+02 (9.7082e+00)	2.1924e+02 (1.0445e+01)
CPSO	8.2179e+02 (7.9673e+00)	1.2179e+03 (9.7679e+01)	5.4617e+02 (5.1238e+01)	9.6411e+02 (5.9474e+00)	9.5683e+02 (5.8653e+01)
Hierarchical D-LPSO	**5.0000e+02 (1.0000e+00)**	**5.0000e+02 (1.2189e+00)**	**5.3416e+02 (1.2212e+00)**	**2.0000e+02 (1.9874e-06)**	**2.1001e+02 (1.5874e+00)**

Table 6: Comparisons Between Hierarchical d-lpso And Other Algorithms on the basis of WilCoxon's Ranksum Tests

Algorithms	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}	f_{21}	f_{22}	f_{23}	f_{24}	f_{25}
DE/rand/1/bin	-1	-1	-1	1	1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
jDE	-1	-1	-1	1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
UPSO	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
CLPSO	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
DMS-PSO	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
wFIPS	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
CPSO	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1(better)							0(equal)									-1(worse)									

The above five comparison tables (Table 1 to Table 5) indicate that out of the 25 in 22 cases Hierarchical D-LPSO has outperformed the competitor algorithms considering mean error. In case of rotated and hybrid composition functions the performance of Hierarchical D-LPSO is quite well. It performs well also in case of functions involving noise in fitness. Hierarchical D-LPSO has been outperformed only in three functions. It can be also seen from the tables that it provides statistically superior results than the other PSO-variants in almost all the functions.

VIII. CONCLUSIONS

After its development more than a decade ago, PSO has eventually become a very powerful method of real-parametric function optimization. The new variant of PSO proposed here and referred to as Hierarchical D-LPSO, has been studied for a set of test functions. The performance of this new algorithm on the benchmark functions of CEC-05 is compared to other existing algorithms like CLPSO, CPSO, jDE, wFIPS, etc. Hierarchical D-LPSO performs very well on all the functions and it has outperformed the competitor algorithms over 22 out of 25 cases in a statistically significant fashion.

Nowadays an extensive research work is going on in designing various algorithms to optimize large-scale high-dimensional problems (D = 1000, D = 500). It needs focused research to improve the performance of Hierarchical D-LPSO in case of high dimensional problems. Our future work will be on the improvement of Hierarchical D-LPSO.

ACKNOWLEDGEMENTS

This work was supported by the Czech Science Foundation, under the grant no. GA102/09/1494.

REFERENCES

[1] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Conf. Neural Networks IV*, Piscataway, NJ, 1995.

[2] R. C. Eberhart and Y. Shi, "Particle swarm optimization: developments, applications and resources," in *Proc. 2001 Congr. Evolutionary Computation*, vol. 1, 2001.

[3] H.-K. Tsai, J.-M. Yang, Y.-F. Tsai, and C.-Y. Kao, "Some issues of designing genetic algorithms for traveling salesman problems," *Soft Comput.*, vol. 8, no. 10, pp. 689–697, Nov. 2004.

[4] K. E. Parsopoulos and M. N. Vrahatis, "UPSO: A unified particle Swarm optimization scheme" In: Lecture Series on Computer and Computational Sciences, Vol. 1, *Proc. Int. Conf. Computational Methods in Sciences and Engineering* (ICCMSE 2004), VSP International Science Publishers, Zeist, The Netherlands (2004) 868–873

[5] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 281–295, Jun. 2006.

[6] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: Simpler, maybe better," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3,pp. 204–210, Jun. 2004

[7] J. J. Liang and P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer," in *Proc. Swarm Intell. Symp.*, Jun. 2005, pp. 124–129.

[8] R. Storn and K. Price, "Differential evolution a simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[9] J. Brest, S. Greiner, B. Bo˘skovi´c, M. Mernik, and V. ˘Zumer, "Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[10] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," *Technical Report, Nanyang Technological University*, Singapore, May 2005 and *KanGAL Report* #2005005, IIT Kanpur, India.

[11] S. Janson and M. Middendorf, "A Hierarchical Particle Swarm Optimizer and Its Adaptive Variant", IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics, vol. 35, no. 6, Dec. 2005.

[12] M. Clerc and J. Kennedy, "The particle swarm–explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, Feb. 2002.

[13] F. Wilcoxon, "Individual comparisons by ranking methods", *Biometrics*, 1, 80-83, 1945.

[14] S. García, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behavior: a case study on the CEC'2005 special session on real parameter optimization", *Journal of Heuristics*, Vol. 15, Issue 6, pp. 617-644 Dec. 2009